

Package ‘groupdata2’

January 28, 2017

Title Creating Groups from Data

Version 0.1.0

Date 2017-01-25

Description Subsetting methods for balanced cross-validation, time series windowing,
and general grouping and splitting of data.

Depends R (>= 3.3.1)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, plyr, utils

RoxygenNote 5.0.1

Suggests ggplot2, knitr, rmarkdown, tidyr, broom, testthat, lmerTest,
hydroGOF

VignetteBuilder knitr

NeedsCompilation no

Author Ludvig Renbo Olsen [aut, cre]

Maintainer Ludvig Renbo Olsen <r-pkgs@ludvigolsen.dk>

Repository CRAN

Date/Publication 2017-01-28 19:16:49

R topics documented:

fold	2
group	3
groupdata2	5
group_factor	6
splt	7
%staircase%	9

Index	10
--------------	-----------

fold	Create balanced folds for cross-validation.
------	---

Description

Divides data into groups by a range of methods. Balances a given categorical variable between folds and keeps (if possible) all data points with the same ID (e.g. participant_id) in the same fold.

Usage

```
fold(data, k = 5, cat_col = NULL, id_col = NULL, method = "n_dist")
```

Arguments

data	Dataframe or Vector
k	Number of folds, fold size, or step size (depending on chosen method) Given as whole numbers or percentage ($0 < n < 1$).
cat_col	Categorical variable to balance between folds. E.g. when predicting a binary variable (a or b), it is necessary to have both represented in every fold N.B. If also passing an id_col, cat_col should be a constant for that ID.
id_col	Factor with IDs. This will be used to keep all rows with an ID in the same fold (if possible). E.g. If we have measured a participant multiple times and want to see the effect of time, we want to have all observations of this participant in the same fold.
method	greedy, n_dist, n_fill, n_last, n_rand, or staircase greedy divides up the data greedily given a specified group size (e.g. 10, 10, 10, 10, 10, 7) n_dist divides the data into a specified number of groups and distributes excess data points across groups (e.g. 11, 11, 12, 11, 12) n_fill divides the data into a specified number of groups and fills up groups with excess data points from the beginning (e.g. 12, 12, 11, 11, 11) n_last divides the data into a specified number of groups. The algorithm finds the most equal group sizes possible, using all data points. Only the last group is able to differ in size (e.g. 11, 11, 11, 11, 13) n_rand divides the data into a specified number of groups. Excess data points are placed randomly in groups (only 1 per group) (e.g. 12, 11, 11, 11, 12) staircase uses step_size to divide up the data. Group size increases with 1 step for every group, until there is no more data (e.g. 5, 10, 15, 20, 7)

Details

cat_col: data is first subset by cat_col. Subsets are folded/grouped and merged. || id_col: folds are created from unique IDs. || cat_col AND id_col: data is subset by cat_col and folds are created from unique IDs in each subset. Subsets are merged.

Value

Dataframe with grouping factor for subsetting in cross-validation.

Author(s)

Ludvig Renbo Olsen, <r-pkgs@ludvigolsen.dk>

Examples

```
# Attach packages
library(groupdata2)
library(dplyr)

# Create dataframe
df <- data.frame(
  "participant" = factor(rep(c('1','2', '3', '4', '5', '6'), 3)),
  "age" = rep(sample(c(1:100), 6), 3),
  "diagnosis" = rep(c('a', 'b', 'a', 'a', 'b', 'b'), 3),
  "score" = sample(c(1:100), 3*6))
df <- df[order(df$participant),]
df$session <- rep(c('1','2', '3'), 6)

# Using fold()
# Without cat_col and id_col
df_folded <- fold(df, 3, method = 'n_dist')

# With cat_col
df_folded <- fold(df, 3, cat_col = 'diagnosis',
  method = 'n_dist')

# With id_col
df_folded <- fold(df, 3, id_col = 'participant',
  method = 'n_dist')

# With cat_col and id_col
df_folded <- fold(df, 3, cat_col = 'diagnosis',
  id_col = 'participant', method = 'n_dist')

# Order by folds
df_folded <- df_folded[order(df_folded$.folds),]
```

group

Create groups from your data.

Description

Divides data into groups by a range of methods. Creates a grouping factor with 1s for group 1, 2s for group 2, etc. Returns a dataframe grouped by the grouping factor for easy use in dplyr pipelines.

Usage

```
group(data, n, method = "n_dist", force_equal = FALSE, allow_zero = FALSE,
      return_factor = FALSE, descending = FALSE, randomize = FALSE,
      col_name = ".groups")
```

Arguments

data	Dataframe or Vector
n	Number of groups, group size, or step size (depending on chosen method) Given as whole numbers or percentage ($0 < n < 1$).
method	greedy, n_dist, n_fill, n_last, n_rand, or staircase greedy divides up the data greedily given a specified group size (<i>e.g.</i> 10, 10, 10, 10, 10, 7) n_dist divides the data into a specified number of groups and distributes excess data points across groups (<i>e.g.</i> 11, 11, 12, 11, 12) n_fill divides the data into a specified number of groups and fills up groups with excess data points from the beginning (<i>e.g.</i> 12, 12, 11, 11, 11) n_last divides the data into a specified number of groups. The algorithm finds the most equal group sizes possible, using all data points. Only the last group is able to differ in size (<i>e.g.</i> 11, 11, 11, 11, 13) n_rand divides the data into a specified number of groups. Excess data points are placed randomly in groups (only 1 per group) (<i>e.g.</i> 12, 11, 11, 11, 12) staircase uses step_size to divide up the data. Group size increases with 1 step for every group, until there is no more data (<i>e.g.</i> 5, 10, 15, 20, 7)
force_equal	Create equal groups by discarding excess data points. Implementation varies between methods. (Logical)
allow_zero	Whether n can be passed as 0. (Logical)
return_factor	Return only grouping factor (Logical)
descending	Change direction of method. (Not fully implemented) (Logical)
randomize	Randomize the grouping factor (Logical)
col_name	Name of added grouping factor

Value

Dataframe grouped by new grouping factor

Author(s)

Ludvig Renbo Olsen, <r-pkgs@ludvigolsen.dk>

See Also

Other grouping functions: [group_factor](#), [split](#)

Examples

```
# Attach packages
library(groupdata2)
library(dplyr)

# Create dataframe
df <- data.frame("x"=c(1:12),
  "species" = rep(c('cat','pig', 'human'), 4),
  "age" = sample(c(1:100), 12))

# Using group()
df_grouped <- group(df, 5, method = 'n_dist')

# Using group() with dplyr pipeline to get mean age
df_means <- df %>%
  group(5, method = 'n_dist') %>%
  dplyr::summarise(mean_age = mean(age))
```

groupdata2

groupdata2: A package for creating groups from data

Description

Subsetting Methods for Balanced Cross-Validation, Time Series Windowing, and General Grouping and Splitting of Data.

Details

The groupdata2 package provides four main functions: `group`, `group_factor`, `splt`, and `fold`

group

Create groups from your data.

Divides data into groups by a range of methods. Creates a grouping factor with 1s for group 1, 2s for group 2, etc. Returns a dataframe grouped by the grouping factor for easy use in dplyr pipelines.

Go to [group](#)

group_factor

Create grouping factor for subsetting your data.

Divides data into groups by a range of methods. Creates and returns a grouping factor with 1s for group 1, 2s for group 2, etc.

Go to [group_factor](#)

splt

Split data by a range of methods.

Divides data into groups by a range of methods. Splits data by these groups.

Go to [splt](#)

fold

Create balanced folds for cross-validation.

Divides data into groups (folds) by a range of methods. Balances a given categorical variable between folds and keeps (if possible) all data points with the same ID (e.g. participant_id) in the same fold.

Go to [fold](#)

Author(s)

Ludvig Renbo Olsen, <mail@ludvigolsen.dk>

group_factor

Create grouping factor for subsetting your data.

Description

Divides data into groups by a range of methods. Creates and returns a grouping factor with 1s for group 1, 2s for group 2, etc.

Usage

```
group_factor(data, n, method = "n_dist", force_equal = FALSE,
  allow_zero = FALSE, descending = FALSE, randomize = FALSE)
```

Arguments

data	Dataframe or Vector
n	Number of groups, group size, or step size (depending on chosen method) Given as whole numbers or percentage ($0 < n < 1$).
method	greedy, n_dist, n_fill, n_last, n_rand, or staircase greedy divides up the data greedily given a specified group size (e.g. 10, 10, 10, 10, 10, 7) n_dist divides the data into a specified number of groups and distributes excess data points across groups (e.g. 11, 11, 12, 11, 12) n_fill divides the data into a specified number of groups and fills up groups with excess data points from the beginning (e.g. 12, 12, 11, 11, 11) n_last divides the data into a specified number of groups. The algorithm finds the most equal group sizes possible, using all data points. Only the last group is able to differ in size (e.g. 11, 11, 11, 11, 13)

	n_rand divides the data into a specified number of groups. Excess data points are placed randomly in groups (only 1 per group) (<i>e.g.</i> 12, 11, 11, 11, 12)
	staircase uses step_size to divide up the data. Group size increases with 1 step for every group, until there is no more data (<i>e.g.</i> 5, 10, 15, 20, 7)
force_equal	Create equal groups by discarding excess data points. Implementation varies between methods. (Logical)
allow_zero	Whether n can be passed as 0. (Logical)
descending	Change direction of method. (Not fully implemented) (Logical)
randomize	Randomize the grouping factor (Logical)

Value

Grouping factor with 1s for group 1, 2s for group 2, etc.

Author(s)

Ludvig Renbo Olsen, <r-pkgs@ludvigolsen.dk>

See Also

Other grouping functions: [group](#), [splt](#)

Other staircase tools: [%staircase%](#)

Examples

```
# Attach packages
library(groupdata2)
library(dplyr)

# Create a dataframe
df <- data.frame("x"=c(1:12),
  "species" = rep(c('cat', 'pig', 'human'), 4),
  "age" = sample(c(1:100), 12))

# Using group_factor()
groups <- group_factor(df, 5, method = 'n_dist')
df$groups <- groups
```

splt

Split data by a range of methods.

Description

Divides data into groups by a range of methods. Splits data by these groups.

Usage

```
splt(data, n, method = "n_dist", force_equal = FALSE, allow_zero = FALSE,
      descending = FALSE, randomize = FALSE)
```

Arguments

data	Dataframe or Vector
n	Number of groups, group size, or step size (depending on chosen method) Given as whole numbers or percentage ($0 < n < 1$).
method	greedy, n_dist, n_fill, n_last, n_rand, or staircase greedy divides up the data greedily given a specified group size (<i>e.g.</i> 10, 10, 10, 10, 10, 7) n_dist divides the data into a specified number of groups and distributes excess data points across groups (<i>e.g.</i> 11, 11, 12, 11, 12) n_fill divides the data into a specified number of groups and fills up groups with excess data points from the beginning (<i>e.g.</i> 12, 12, 11, 11, 11) n_last divides the data into a specified number of groups. The algorithm finds the most equal group sizes possible, using all data points. Only the last group is able to differ in size (<i>e.g.</i> 11, 11, 11, 11, 13) n_rand divides the data into a specified number of groups. Excess data points are placed randomly in groups (only 1 per group) (<i>e.g.</i> 12, 11, 11, 11, 12) staircase uses step_size to divide up the data. Group size increases with 1 step for every group, until there is no more data (<i>e.g.</i> 5, 10, 15, 20, 7)
force_equal	Create equal groups by discarding excess data points. Implementation varies between methods. (Logical)
allow_zero	Whether n can be passed as 0. (Logical)
descending	Change direction of method. (Not fully implemented) (Logical)
randomize	Randomize the grouping factor (Logical)

Value

List of splitted data

Author(s)

Ludvig Renbo Olsen, <r-pkgs@ludvigolsen.dk>

See Also

Other grouping functions: [group_factor](#), [group](#)

Examples

```
# Attach packages
library(groupdata2)
library(dplyr)

# Create dataframe
```



```
df <- data.frame("x"=c(1:12),
  "species" = rep(c('cat', 'pig', 'human'), 4),
  "age" = sample(c(1:100), 12))

# Using splt()
df_list <- splt(df, 5, method = 'n_dist')
```

%staircase%*Find remainder from staircase method.*

Description

When using the staircase method, the last group might not have the size of the second last group + step size. Use %staircase% to find this remainder.

Usage

```
size %staircase% step_size
```

Arguments

size	Size to staircase (Integer)
step_size	Step size (Integer)

Value

Remainder (Integer). Returns 0 if the last group has the size of the second last group + step size.

Author(s)

Ludvig Renbo Olsen, <mail@ludvigolsen.dk>

See Also

Other staircase tools: [group_factor](#)

Examples

```
# Attach packages
library(groupdata2)

100 %staircase% 2

# Finding remainder with value 0
size = 150
for (step_size in c(1:30)){
  if(size %staircase% step_size == 0){
    print(step_size)
  }
}
```

Index

`%staircase%`, [7](#), [9](#)

`binning (group)`, [3](#)

`fold`, [2](#), [6](#)

`group`, [3](#), [5](#), [7](#), [8](#)

`group_factor`, [4](#), [5](#), [6](#), [8](#), [9](#)

`groupdata2`, [5](#)

`groupdata2-package (groupdata2)`, [5](#)

`remainder (%staircase%)`, [9](#)

`split (group)`, [3](#)

`splt`, [4](#), [6](#), [7](#), [7](#)

`staircase (%staircase%)`, [9](#)

`window (group)`, [3](#)