Package ‘strapgod’

September 20, 2019

Title  Resampled Data Frames
Version  0.0.4
Description  Create data frames with virtual groups that can be used with
’dplyr’ to efficiently compute resampled statistics, generate the data for
hypothetical outcome plots, and fit multiple models on resampled variations
of the original data.
License  GPL-3
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1
Depends  R (>= 3.2.0)
Imports  dplyr (>= 0.8.3), tidyr (>= 1.0.0), rlang, tibble, lifecycle
Suggests  broom, testthat, covr, knitr, rmarkdown
URL  https://github.com/DavisVaughan/strapgod
BugReports  https://github.com/DavisVaughan/strapgod/issues
VignetteBuilder  knitr
NeedsCompilation  no
Author  Davis Vaughan [aut, cre]
Maintainer  Davis Vaughan <davis@rstudio.com>
Repository  CRAN
Date/Publication  2019-09-20 04:50:02 UTC

R topics documented:

  bootstrapify .......................................................... 2
  collect.resampled_df ............................................. 3
  samplify ............................................................ 4

Index  ................................................................. 7
bootstrify

Create a bootstrapped tibble

Description

bootstrify() creates a bootstrapped tibble with virtual groups.

Usage

bootstrify(data, times, ..., key = ".bootstrap")

Arguments

data: A tbl.
times: A single integer specifying the number of resamples. If the tibble is grouped, this is the number of resamples per group.
...: Not used.
key: A single character specifying the name of the virtual group that is added.

Details

The following functions have special / interesting behavior when used with a resampled_df:

- dplyr::collect()
- dplyr::summarise()
- dplyr::do()
- dplyr::group_map()
- dplyr::group_modify()
- dplyr::group_walk()
- dplyr::group_nest()
- dplyr::group_split()

Value

A resampled_df with an extra group specified by the key.

See Also

collect.resampled_df()

Other virtual samplers: samplify
Examples

```r
library(dplyr)
library(broom)

bootstrapify(iris, 5)

iris %>%
  bootstrapify(5) %>%
  summarise(per_strap_mean = mean(Petal.Width))

iris %>%
  group_by(Species) %>%
  bootstrapify(5) %>%
  summarise(per_strap_species_mean = mean(Petal.Width))

iris %>%
  bootstrapify(5) %>%
  do(tidy(lm(Sepal.Width ~ Sepal.Length + Species, data = .)))

# Alternatively, use the newer group_modify()

iris %>%
  bootstrapify(5) %>%
  group_modify(~tidy(lm(Sepal.Width ~ Sepal.Length + Species, data = .x)))

# Alter the name of the group with 'key'
# Materialize them with collect()
straps <- bootstrapify(iris, 5, key = ".straps")
collect(straps)
```

---

**collect.resampled_df**  
*Force virtual groups to become explicit rows*

**Description**

When `collect()` is used on a `resampled_df`, the virtual bootstrap groups are made explicit.

**Usage**

```r
## S3 method for class 'resampled_df'
collect(x, ..., id = NULL, original_id = NULL)
```

**Arguments**

- `x`  
  A `resampled_df`.
- `...`  
  Not used.
- `id`  
  Optional. A single character that specifies a name for a column containing a sequence from 1:n for each bootstrap group.
- `original_id`  
  Optional. A single character that specifies a name for a column containing the original position of the bootstrapped row.
Examples

```r
library(dplyr)

# virtual groups become real rows
collect(bootstrapify(iris, 5))

# add on the id column for an identifier per bootstrap
collect(bootstrapify(iris, 5), id = "\.id")

# add on the original_id column to know which row this bootstrapped row
# originally came from
collect(bootstrapify(iris, 5), original_id = "\.original_id")
```

---

**samplify**  
*Created a resampled tibble*

---

**Description**

`samplify()` creates a resampled tibble with *virtual groups*.

**Usage**

```r
samplify(data, times, size, ..., replace = FALSE, key = "\.sample")
```

**Arguments**

- `data`  
  A tbl.

- `times`  
  A single integer specifying the number of resamples. If the tibble is grouped, this is the number of resamples per group.

- `size`  
  A single integer specifying the size of each resample. For a grouped data frame, this is also allowed to be an integer vector with size equal to the number of groups in `data`. This can be helpful when sampling without replacement when the number of rows per group is very different.

- `...`  
  Not used.

- `replace`  
  Whether or not to sample with replacement.

- `key`  
  A single character specifying the name of the virtual group that is added.

**Details**

The following functions have special / interesting behavior when used with a resampled_df:

- `dplyr::collect()`  
- `dplyr::summarise()`  
- `dplyr::do()`  
- `dplyr::group_map()`
samplify

- `dplyr::group_modify()`
- `dplyr::group_walk()`
- `dplyr::group_nest()`
- `dplyr::group_split()`

Value

A resampled_df with an extra group specified by the key.

See Also

`collect.resampled_df()`

Other virtual samplers: `bootstrapify`

Examples

```r
library(dplyr)
library(broom)

samplify(iris, times = 3, size = 20)

iris %>%
  samplify(times = 3, size = 20) %>%
  summarise(per_strap_mean = mean(Petal.Width))

iris %>%
  group_by(Species) %>%
  samplify(times = 3, size = 20) %>%
  summarise(per_strap_species_mean = mean(Petal.Width))

# Alter the name of the group with `key`
# Materialize them with collect()
samps <- samplify(iris, times = 3, size = 5, key = ".samps")
collect(samps)

collect(samps, id = ".id", original_id = ".orig_id")

# Be careful not to specify a `size` larger
# than one of your groups! This will throw an error.

iris_group_sizes_of_50_and_5 <- iris[1:55, ] %>%
  group_by(Species) %>%
  group_trim()

count(iris_group_sizes_of_50_and_5, Species)

# size = 10 > min_group_size = 5
## Not run:
iris_group_sizes_of_50_and_5 %>%

```
samplify(times = 2, size = 10)

## End(Not run)

# Instead, pass a vector of sizes to `samplify()` if this
# structure is absolutely required for your use case.

# size of 10 for the first group
# size of 5 for the second group
# total number of rows is 10 * 2 + 5 * 2 = 30
iris_group_sizes_of_50_and_5 %>%
  samplify(times = 2, size = c(10, 5)) %>%
collect()
Index

bootstrapify, 2, 5

collect.resampled_df, 3
collect.resampled_df(), 2, 5

dplyr::collect(), 2, 4
dplyr::do(), 2, 4
dplyr::group_map(), 2, 4
dplyr::group_modify(), 2, 5
dplyr::group_nest(), 2, 5
dplyr::group_split(), 2, 5
dplyr::group_walk(), 2, 5
dplyr::summarise(), 2, 5

samplify, 2, 4