Package ‘rsample’

July 13, 2019

Title  General Resampling Infrastructure
Version 0.0.5
Maintainer Max Kuhn <max@rstudio.com>
Description Classes and functions to create and summarize different types of resampling objects (e.g. bootstrap, cross-validation).
Imports dplyr, purrr, tibble, rlang (>= 0.4.0), methods, generics, utils, tidyselect, furrr
Depends R (>= 3.1), tidy
Suggests ggplot2, testthat, rmarkdown, knitr, AmesHousing, recipes (>= 0.1.4), broom
URL https://tidymodels.github.io/rsample
BugReports https://github.com/tidymodels/rsample/issues
License GPL-2
Encoding UTF-8
VignetteBuilder knitr
LazyData true
RoxygenNote 6.1.1
NeedsCompilation no
Author Max Kuhn [aut, cre], Fanny Chow [aut], Hadley Wickham [aut], RStudio [cph]
Repository CRAN
Date/Publication 2019-07-12 22:20:11 UTC

R topics documented:

  add_resample_id .......................................................... 2
  apparent ................................................................. 3
add_resample_id

Augment a data set with resampling identifiers

Description

For a data set, add_resample_id() will add at least one new column that identifies which resample that the data came from. In most cases, a single column is added but for some resampling methods two or more are added.

Usage

add_resample_id(.data, split, dots = FALSE)

Arguments

.data A data frame
.split A single rset object.
dots A single logical: should the id columns be prefixed with a "." to avoid name conflicts with .data?
Sampling for the Apparent Error Rate

Description

When building a model on a data set and re-predicting the same data, the performance estimate from those predictions is often called the “apparent” performance of the model. This estimate can be wildly optimistic. "Apparent sampling" here means that the analysis and assessment samples are the same. These resamples are sometimes used in the analysis of bootstrap samples and should otherwise be avoided like old sushi.

Usage

apparent(data, ...)

Arguments

data A data frame.

... Not currently used.

Value

A tibble with a single row and classes apparent, rset, tbl_df, tbl, and data.frame. The results include a column for the data split objects and one column called id that has a character string with the resample identifier.
as.data.frame.rsplit

Convert an rsplit object to a data frame

Description

The analysis or assessment code can be returned as a data frame (as dictated by the data argument) using as.data.frame.rsplit. analysis and assessment are shortcuts.

Usage

```r
## S3 method for class 'rsplit'
as.data.frame(x, row.names = NULL, optional = FALSE,
             data = "analysis", ...)

analysis(x, ...)

assessment(x, ...)
```

Arguments

- **x**: An rsplit object.
- **row.names**: NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
- **optional**: A logical: should the column names of the data be checked for legality?
- **data**: Either "analysis" or "assessment" to specify which data are returned.
- **...**: Additional arguments to be passed to or from methods. Not currently used.

Examples

```r
library(dplyr)
set.seed(104)
folds <- vfold_cv(mtcars)

model_data_1 <- folds$splits[[1]] %>% analysis()
holdout_data_1 <- folds$splits[[1]] %>% assessment()
```
Description

Job Attrition

Details

These data are from the IBM Watson Analytics Lab. The website describes the data with “Uncover the factors that lead to employee attrition and explore important questions such as ‘show me a breakdown of distance from home by job role and attrition’ or ‘compare average monthly income by education and attrition’. This is a fictional data set created by IBM data scientists.”. There are 1470 rows.

Value

attrition       a data frame

Source

The IBM Watson Analytics Lab website https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/

Examples

data(attrition)
str(attrition)

Description

A bootstrap sample is a sample that is the same size as the original data set that is made using replacement. This results in analysis samples that have multiple replicates of some of the original rows of the data. The assessment set is defined as the rows of the original data that were not included in the bootstrap sample. This is often referred to as the “out-of-bag” (OOB) sample.

Usage

bootstraps(data, times = 25, strata = NULL, breaks = 4, apparent = FALSE, ...)

Arguments

- **data**: A data frame.
- **times**: The number of bootstrap samples.
- **strata**: A variable that is used to conduct stratified sampling. When not NULL, each bootstrap sample is created within the stratification variable. This could be a single character value or a variable name that corresponds to a variable that exists in the data frame.
- **breaks**: A single number giving the number of bins desired to stratify a numeric stratification variable.
- **apparent**: A logical. Should an extra resample be added where the analysis and holdout subset are the entire data set. This is required for some estimators used by the summary function that require the apparent error rate.
- ... Not currently used.

Details

The argument apparent enables the option of an additional "resample" where the analysis and assessment data sets are the same as the original data set. This can be required for some types of analysis of the bootstrap results. The strata argument is based on a similar argument in the random forest package where the bootstrap samples are conducted **within the stratification variable**. This can help ensure that the number of data points in the bootstrap sample is equivalent to the proportions in the original data set.

Value

An tibble with classes bootstraps, rset, tbl_df, tbl, and data.frame. The results include a column for the data split objects and a column called id that has a character string with the resample identifier.

Examples

```r
bootstraps(mtcars, times = 2)
bootstraps(mtcars, times = 2, apparent = TRUE)

library(purrr)
iris2 <- iris[1:130, ]
set.seed(13)
resample1 <- bootstraps(iris2, times = 3)
map_dbl(resample1$splits, 
    function(x) {
        dat <- as.data.frame(x)$Species
        mean(dat == "virginica")
    })

set.seed(13)
resample2 <- bootstraps(iris2, strata = "Species", times = 3)
map_dbl(resample2$splits, 
    function(x) {
...
complement

```r
dat <- as.data.frame(x)$Species
mean(dat == "virginica")
```

```r
set.seed(13)
resample3 <- bootstrap(iris, strata = "Sepal.Length", breaks = 6, times = 3)
map_dbl(resample3$splits,
function(x) {
  dat <- as.data.frame(x)$Species
  mean(dat == "virginica")
})
```

---

**complement**

*Determine the Assessment Samples*

**Description**

Given an `rsplit` object, `complement` will determine which of the data rows are contained in the assessment set. To save space, many of the `rset` objects will not contain indices for the assessment split.

**Usage**

```r
complement(x, ...)
```

**Arguments**

- **x**
  - An `rsplit` object

- **...**
  - Not currently used

**Value**

A integer vector.

**See Also**

`populate()`

**Examples**

```r
set.seed(28432)
fold_rs <- vfold_cv(mtcars)
head(fold_rs$splits[[1]]$in_id)
fold_rs$splits[[1]]$out_id
complement(fold_rs$splits[[1]])
```
**drinks**  
*Sample Time Series Data*

**Description**

Sample Time Series Data

**Details**

Drink sales. The exact name of the series from FRED is: "Merchant Wholesalers, Except Manufacturers’ Sales Branches and Offices Sales: Nondurable Goods: Beer, Wine, and Distilled Alcoholic Beverages Sales"

**Value**

| drinks | a data frame |

**Source**

The Federal Reserve Bank of St. Louis website https://fred.stlouisfed.org/series/S4248SM144NCEN

**Examples**

```r
data(drinks)
str(drinks)
```

---

**form_pred**  
*Extract Predictor Names from Formula or Terms*

**Description**

`all.vars` returns all variables used in a formula. This function only returns the variables explicitly used on the right-hand side (i.e., it will not resolve dots unless the object is `terms` with a data set specified).

**Usage**

`form_pred(object, ...)`

**Arguments**

- `object`  
  A model formula or `stats::terms()` object.
- `...`  
  Arguments to pass to `all.vars()`

**Value**

A character vector of names
Examples

form_pred(y ~ x + z)
form_pred(terms(y ~ x + z))

form_pred(y ~ x + log(z))
form_pred(log(y) ~ x + z)

form_pred(y1 + y2 ~ x + z)
form_pred(log(y1) + y2 ~ x + z)

# will fail:
# form_pred(y ~ .)

form_pred(terms(Species ~ (.)*2, data = iris))
form_pred(terms( ~ (.)*2, data = iris))

---

gather.rset  Gather an rset Object

Description

This method uses gather on an rset object to stack all of the non-ID or split columns in the data and is useful for stacking model evaluation statistics. The resulting data frame has a column based on the column names of data and another for the values.

Usage

## S3 method for class 'rset'
gather(data, key = NULL, value = NULL, ...,
       na.rm = TRUE, convert = FALSE, factor_key = TRUE)

Arguments

data  An rset object.
key, value, ...  Not specified in this method and will be ignored. Note that this means that selectors are ignored if they are passed to the function.
na.rm  If TRUE, will remove rows from output where the value column in NA.
convert  If TRUE will automatically run type.convert() on the key column. This is useful if the column names are actually numeric, integer, or logical.
factor_key  If FALSE, the default, the key values will be stored as a character vector. If TRUE, will be stored as a factor, which preserves the original ordering of the columns.

Value

A data frame with the ID columns, a column called model (with the previous column names), and a column called statistic (with the values).
Examples

```r
library(rsample)
cv_obj <- vfold_cv(mtcars, v = 10)
cv_obj$lm_rmse <- rnorm(10, mean = 2)
cv_obj$nnnet_rmse <- rnorm(10, mean = 1)
gather(cv_obj)
```

Description

Group V-fold cross-validation creates splits of the data based on some grouping variable (which may have more than a single row associated with it). The function can create as many splits as there are unique values of the grouping variable or it can create a smaller set of splits where more than one value is left out at a time.

Usage

```r
group_vfold_cv(data, group = NULL, v = NULL, ...)
```

Arguments

data A data frame.
group This could be a single character value or a variable name that corresponds to a variable that exists in the data frame.
v The number of partitions of the data set. If let NULL, v will be set to the number of unique values in the group.
... Not currently used.

Value

An tibble with classes group_vfold_cv, rset, tbl_df, tbl, and data.frame. The results include a column for the data split objects and an identification variable.

Examples

```r
set.seed(3527)
test_data <- data.frame(id = sort(sample(1:20, size = 80, replace = TRUE)))
test_data$dat <- runif(nrow(test_data))

set.seed(5144)
split_by_id <- group_vfold_cv(test_data, group = "id")

get_id_left_out <- function(x)
unique(assessment(x)$id)
```
library(purrr)
table(map_int(split_by_id$splits, get_id_left_out))

set.seed(5144)
split_by_some_id <- group_vfold_cv(test_data, group = "id", v = 7)
held_out <- map(split_by_some_id$splits, get_id_left_out)
table(unlist(held_out))
# number held out per resample:
map_int(held_out, length)

---

### initial_split

**Simple Training/Test Set Splitting**

**Description**

`initial_split` creates a single binary split of the data into a training set and testing set. `initial_time_split` does the same, but takes the first `prop` samples for training, instead of a random selection. Training and testing are used to extract the resulting data.

**Usage**

```r
initial_split(data, prop = 3/4, strata = NULL, breaks = 4, ...)
initial_time_split(data, prop = 3/4, ...)
training(x)
testing(x)
```

**Arguments**

- **data**: A data frame.
- **prop**: The proportion of data to be retained for modeling/analysis.
- **strata**: A variable that is used to conduct stratified sampling to create the resamples. This could be a single character value or a variable name that corresponds to a variable that exists in the data frame.
- **breaks**: A single number giving the number of bins desired to stratify a numeric stratification variable.
- **...**: Not currently used.
- **x**: An rsplit object produced by `initial_split`

**Details**

The `strata` argument causes the random sampling to be conducted *within the stratification variable*. The can help ensure that the number of data points in the training data is equivalent to the proportions in the original data set.
Value

An rset object that can be used with the training and testing functions to extract the data in each split.

Examples

```r
set.seed(1353)
car_split <- initial_split(mtcars)
train_data <- training(car_split)
test_data <- testing(car_split)

drinks_split <- initial_time_split(drinks)
train_data <- training(drinks_split)
test_data <- testing(car_split)
c(max(train_data$date, min(test_data$date)) # no overlap
```

### int_pctl

**Bootstrap confidence intervals**

**Description**

Calculate bootstrap confidence intervals using various methods.

**Usage**

```r
int_pctl(.data, statistics, alpha = 0.05)
int_t(.data, statistics, alpha = 0.05)
int_bca(.data, statistics, alpha = 0.05, .fn, ...)
```

**Arguments**

- `.data` A data frame containing the bootstrap resamples created using bootstraps(). For t- and BCa-intervals, the apparent argument should be set to TRUE.
- `statistics` An unquoted column name or dplyr selector that identifies a single column in the data set that contains the individual bootstrap estimates. This can be a list column of tidy tibbles (that contains columns term and estimate) or a simple numeric column. For t-intervals, a standard tidy column (usually called `std.err`) is required. See the examples below.
- `alpha` Level of significance
- `.fn` A function to calculate statistic of interest. The function should take an rsplit as the first argument and the ... are required.
- `...` Arguments to pass to `.fn`. 

Details

Percentile intervals are the standard method of obtaining confidence intervals but require thousands of resamples to be accurate. t-intervals may need fewer resamples but require a corresponding variance estimate. Bias-corrected and accelerated intervals require the original function that was used to create the statistics of interest and are computationally taxing.

Value

Each function returns a tibble with columns .lower, .estimate, .upper, .alpha, .method, and term. .method is the type of interval (eg. "percentile", "student-t", or "BCa"). term is the name of the estimate.

References

https://tidymodels.github.io/rsample/articles/Applications/Intervals.html

Examples

```r
library(broom)
library(dplyr)
library(purrr)
library(tibble)

lm_est <- function(split) {
  lm(mpg ~ disp + hp, data = analysis(split)) %>%
    tidy()
}

set.seed(52156)
car_rs <-
  bootstraps(mtcars, 1000, apparent = TRUE) %>%
  mutate(results = map(splits, lm_est))

int_pctl(car_rs, results)
int_t(car_rs, results)
int_bca(car_rs, results, .fn = lm_est)

# putting results into a tidy format
rank_corr <- function(split) {
  dat <- analysis(split)
  tibble(
    term = "corr",
    estimate = cor(dat$Sepal.Length, dat$Sepal.Width, method = "spearman"),
    # don't know the analytical std.err so no t-intervals
    std.err = NA_real_
  )
}

set.seed(69325)
```
bootstraps(iris, 1000, apparent = TRUE) %>%
  mutate(correlations = map(splits, rank_corr)) %>%
  int_pctl(correlations)

---

### labels.rset

**Find Labels from rset Object**

**Description**

Produce a vector of resampling labels (e.g. "Fold1") from an `rset` object. Currently, `nested_cv` is not supported.

**Usage**

```r
## S3 method for class 'rset'
labels(object, make_factor = FALSE, ...)
```

```r
## S3 method for class 'vfold_cv'
labels(object, make_factor = FALSE, ...)
```

**Arguments**

- `object`: An `rset` object
- `make_factor`: A logical for whether the results should be character or a factor.
- `...`: Not currently used.

**Value**

A single character or factor vector.

**Examples**

```r
labels(vfold_cv(mtcars))
```

---

### labels.rsplit

**Find Labels from rsplit Object**

**Description**

Produce a tibble of identification variables so that single splits can be linked to a particular resample.

**Usage**

```r
## S3 method for class 'rsplit'
labels(object, ...)
```
Arguments

- **object**: An rsplit object
- ... Not currently used.

Value

A tibble.

See Also

- add_resample_id

Examples

```r
cv_splits <- vfold_cv(mtcars)
labels(cv_splits$splits[[1]])
```

---

### loo_cv

**Leave-One-Out Cross-Validation**

**Description**

Leave-one-out (LOO) cross-validation uses one data point in the original set as the assessment data and all other data points as the analysis set. A LOO resampling set has as many resamples as rows in the original data set.

**Usage**

```r
loo_cv(data, ...)
```

**Arguments**

- **data**: A data frame.
- ... Not currently used.

**Value**

An tibble with classes `loo_cv`, `rset`, `tbl_df`, `tbl`, and `data.frame`. The results include a column for the data split objects and one column called `id` that has a character string with the resample identifier.

**Examples**

```r
loo_cv(mtcars)
```
make_strata  

Create or Modify Stratification Variables

Description

For stratified resampling, this function can create strata from numeric data and also make non-numeric data more conducive to be used for stratification.

Usage

```
make_strata(x, breaks = 4, nunique = 5, pool = 0.15, depth = 20)
```

Arguments

- **x**: An input vector.
- **breaks**: A single number giving the number of bins desired to stratify a numeric stratification variable.
- **nunique**: An integer for the number of unique value threshold in the algorithm.
- **pool**: A proportion of data used to determine if a particular group is too small and should be pooled into another group.
- **depth**: An integer that is used to determine the best number of percentiles that should be used. The number of bins are based on \( \min(5, \lfloor n / \text{depth} \rfloor) \) where \( n = \text{length}(x) \). If \( x \) is numeric, there must be at least 40 rows in the data set (when \( \text{depth} = 20 \)) to conduct stratified sampling.

Details

For numeric data, if the number of unique levels is less than \( nunique \), the data are treated as categorical data.

For categorical inputs, the function will find levels of \( x \) than occur in the data with percentage less than \( pool \). The values from these groups will be randomly assigned to the remaining strata (as will data points that have missing values in \( x \)).

For numeric data with more unique values than \( nunique \), the data will be converted to being categorical based on percentiles of the data. The percentile groups will have no more than 20 percent of the data in each group. Again, missing values in \( x \) are randomly assigned to groups.

Value

A factor vector.
**mc_cv**

### Examples

```r
set.seed(61)
x1 <- rpois(100, lambda = 5)
table(x1)
table(make_strata(x1))

set.seed(554)
x2 <- rpois(100, lambda = 1)
table(x2)
table(make_strata(x2))

# small groups are randomly assigned
x3 <- factor(x2)
table(x3)
table(make_strata(x3))

# `oilType` data from `caret`
x4 <- rep(LETTERS[1:7], c(37, 26, 3, 7, 11, 10, 2))
table(x4)
table(make_strata(x4))
table(make_strata(x4, pool = 0.1))
table(make_strata(x4, pool = 0.0))

# not enough data to stratify
x5 <- rnorm(20)
table(make_strata(x5))

set.seed(483)
x6 <- rnorm(200)
quantile(x6, probs = (0:10)/10)
table(make_strata(x6, breaks = 10))
```

---

**mc_cv**  
**Monte Carlo Cross-Validation**

### Description

One resample of Monte Carlo cross-validation takes a random sample (without replacement) of the original data set to be used for analysis. All other data points are added to the assessment set.

### Usage

```r
mc_cv(data, prop = 3/4, times = 25, strata = NULL, breaks = 4, ...)
```

### Arguments

- **data**: A data frame.
- **prop**: The proportion of data to be retained for modeling/analysis.
- **times**: The number of times to repeat the sampling.
strata A variable that is used to conduct stratified sampling to create the resamples. This could be a single character value or a variable name that corresponds to a variable that exists in the data frame.

breaks A single number giving the number of bins desired to stratify a numeric stratification variable.

... Not currently used.

Details

The `strata` argument causes the random sampling to be conducted within the stratification variable. The can help ensure that the number of data points in the analysis data is equivalent to the proportions in the original data set.

Value

An tibble with classes `mc_cv`, `rset`, `tbl_df`, `tbl`, and `data.frame`. The results include a column for the data split objects and a column called `id` that has a character string with the resample identifier.

Examples

```r
mc_cv(mtcars, times = 2)
mc_cv(mtcars, prop = .5, times = 2)

library(purrr)
iris2 <- iris[1:130, ]

set.seed(13)
resample1 <- mc_cv(iris2, times = 3, prop = .5)
map_dbl(resample1$splits,
  function(x) {
    dat <- as.data.frame(x)$Species
    mean(dat == "virginica")
  })

set.seed(13)
resample2 <- mc_cv(iris2, strata = "Species", times = 3, prop = .5)
map_dbl(resample2$splits,
  function(x) {
    dat <- as.data.frame(x)$Species
    mean(dat == "virginica")
  })

set.seed(13)
resample3 <- mc_cv(iris2, strata = "Sepal.Length", breaks = 6, times = 3, prop = .5)
map_dbl(resample3$splits,
  function(x) {
    dat <- as.data.frame(x)$Species
    mean(dat == "virginica")
  })
```
Description

`nested_cv` can be used to take the results of one resampling procedure and conduct further resamples within each split. Any type of resampling used in `rsample` can be used.

Usage

`nested_cv(data, outside, inside)`

Arguments

- `data` A data frame.
- `outside` The initial resampling specification. This can be an already created object or an expression of a new object (see the examples below). If the latter is used, the `data` argument does not need to be specified and, if it is given, will be ignored.
- `inside` An expression for the type of resampling to be conducted within the initial procedure.

Details

It is a bad idea to use bootstrapping as the outer resampling procedure (see the example below)

Value

An tibble with classe `nested_cv` and any other classes that outer resampling process normally contains. The results include a column for the outer data split objects, one or more id columns, and a column of nested tibbles called `inner_resamples` with the additional resamples.

Examples

```r
## Using expressions for the resampling procedures:
nested_cv(mtcars, outside = vfold_cv(v = 3), inside = bootstraps(times = 5))

## Using an existing object:
folds <- vfold_cv(mtcars)
nested_cv(mtcars, folds, inside = bootstraps(times = 5))

## The dangers of outer bootstraps:
set.seed(2222)
bad_idea <- nested_cv(mtcars, outside = bootstraps(times = 5),
                       inside = vfold_cv(v = 3))

first_outer_split <- bad_idea$splits[[1]]
outer_analysis <- as.data.frame(first_outer_split)
```
sum(grepl("Volvo 142E", rownames(outer_analysis)))

## For the 3-fold CV used inside of each bootstrap, how are the replicated
## 'Volvo 142E` data partitioned?
first_inner_split <- bad_idea$inner_resamples[[1]]$splits[[1]]
inner_analysis <- as.data.frame(first_inner_split)
inner_assess <- as.data.frame(first_inner_split, data = "assessment")

sum(grepl("Volvo 142E", rownames(inner_analysis)))
sum(grepl("Volvo 142E", rownames(inner_assess)))

---

**populate**

---

## Description

Many rsplit and rset objects do not contain indicators for the assessment samples. `populate()` can be used to fill the slot for the appropriate indices.

## Usage

`populate(x, ...)`

## Arguments

- **x**: A rsplit and rset object.
- **...**: Not currently used

## Value

An object of the same kind with the integer indicies.

## Examples

```r
code
```

```r
set.seed(28432)
fold_rs <- vfold_cv(mtcars)
fold_rs$splits[[1]]$out_id
complement(fold_rs$splits[[1]])
populate(fold_rs$splits[[1]]$out_id)
fold_rs_all <- populate(fold_rs)
fold_rs_all$splits[[1]]$out_id
```
Description

This resampling method is useful when the data set has a strong time component. The resamples are not random and contain data points that are consecutive values. The function assumes that the original data set are sorted in time order.

Usage

rolling_origin(data, initial = 5, assess = 1, cumulative = TRUE, skip = 0, ...)

Arguments

data: A data frame.
initial: The number of samples used for analysis/modeling in the initial resample.
assess: The number of samples used for each assessment resample.
cumulative: A logical. Should the analysis resample grow beyond the size specified by initial at each resample?.
skip: A integer indicating how many (if any) additional resamples to skip to thin the total amount of data points in the analysis resample. See the example below.
...: Not currently used.

Details

The main options, initial and assess, control the number of data points from the original data that are in the analysis and assessment set, respectively. When cumulative = TRUE, the analysis set will grow as resampling continues while the assessment set size will always remain static. skip enables the function to not use every data point in the resamples. When skip = 0, the resampling data sets will increment by one position. Suppose that the rows of a data set are consecutive days. Using skip = 6 will make the analysis data set operate on weeks instead of days. The assessment set size is not affected by this option.

Value

An tibble with classes rolling_origin, rset, tbl_df, tbl, and data.frame. The results include a column for the data split objects and a column called id that has a character string with the resample identifier.
Examples

```r
set.seed(1131)
ex_data <- data.frame(row = 1:20, some_var = rnorm(20))
dim(rolling_origin(ex_data))
dim(rolling_origin(ex_data, skip = 2))
dim(rolling_origin(ex_data, skip = 2, cumulative = FALSE))

# You can also roll over calendar periods by first nesting by that period,
# which is especially useful for irregular series where a fixed window
# is not useful. This example slides over 5 years at a time.
library(dplyr)
data(drinks)

drinks_annual <- drinks %>%
  mutate(year = as.POSIXlt(date)$year + 1900) %>%
  nest(~year)

multi_year_roll <- rolling_origin(drinks_annual, cumulative = FALSE)

analysis(multi_year_roll$splits[[1]])
assessment(multi_year_roll$splits[[1]])
```

---

rsample

rsample: General Resampling Infrastructure for R

Description

rsample has functions to create variations of a data set that can be used to evaluate models or to estimate the sampling distribution of some statistic.

Terminology

- **A resample** is the result of a two-way split of a data set. For example, when bootstrapping, one part of the resample is a sample with replacement of the original data. The other part of the split contains the instances that were not contained in the bootstrap sample. The data structure rsplit is used to store a single resample.
- When the data are split in two, the portion that are used to estimate the model or calculate the statistic is called the **analysis** set here. In machine learning this is sometimes called the "training set" but this would be poorly named since it might conflict with any initial split of the original data.
- Conversely, the other data in the split are called the **assessment** data. In bootstrapping, these data are often called the "out-of-bag" samples.
- A collection of resamples is contained in an rset object.

Basic Functions

The main resampling functions are: `vfold_cv()`, `bootstraps()`, `mc_cv()`, `rolling_origin()`, and `nested_cv()`.
rsample2caret

Convert Resampling Objects to Other Formats

Description

These functions can convert resampling objects between rsample and caret.

Usage

rsample2caret(object, data = c("analysis", "assessment"))
caret2rsample(ctrl, data = NULL)

Arguments

- object: An rset object. Currently, nested_cv is not supported.
- data: The data that was originally used to produce the ctrl object.
- ctrl: An object produced by trainControl that has had the index and indexOut elements populated by integers. One method of getting this is to extract the control objects from an object produced by train.

Value

rsample2caret returns a list that mimics the index and indexOut elements of a trainControl object. caret2rsample returns an rset object of the appropriate class.

tidy.rsplit

Tidy Resampling Object

Description

The tidy function from the broom package can be used on rset and rsplit objects to generate tibbles with which rows are in the analysis and assessment sets.

Usage

## S3 method for class 'rsplit'
tidy(x, unique_ind = TRUE, ...)

## S3 method for class 'rset'
tidy(x, ...)

## S3 method for class 'vfold_cv'
tidy(x, ...)

## S3 method for class 'nested_cv'
tidy(x, ...)
Arguments

- **x**: A rset or rsplit object
- **unique_ind**: Should unique row identifiers be returned? For example, if FALSE then bootstrapping results will include multiple rows in the sample for the same row in the original data.

Details

Note that for nested resampling, the rows of the inner resample, named `inner_Row`, are relative row indices and do not correspond to the rows in the original data set.

Value

A tibble with columns `Row` and `Data`. The latter has possible values "Analysis" or "Assessment". For `rset` inputs, identification columns are also returned but their names and values depend on the type of resampling. `vfold_cv` contains a column "Fold" and, if repeats are used, another called "Repeats". `bootstraps` and `mc_cv` use the column "Resample".

Examples

```r
library(ggplot2)
theme_set(theme_bw())
set.seed(4121)

cv <- tidy(vfold_cv(mtcars, v = 5))
ggplot(cv, aes(x = Fold, y = Row, fill = Data)) +
  geom_tile() + scale_fill_brewer()

set.seed(4121)

cv <- tidy(vfold_cv(mtcars, v = 5, repeats = 2))
ggplot(cv, aes(x = Fold, y = Row, fill = Data)) +
  geom_tile() + facet_wrap(~Repeat) + scale_fill_brewer()

set.seed(4121)

mc_cv <- tidy(mc_cv(mtcars, times = 5))
ggplot(mc_cv, aes(x = Resample, y = Row, fill = Data)) +
  geom_tile() + scale_fill_brewer()

set.seed(4121)

bt <- tidy(bootstraps(mtcars, time = 5))
ggplot(bt, aes(x = Resample, y = Row, fill = Data)) +
  geom_tile() + scale_fill_brewer()

dat <- data.frame(day = 1:30)
# Resample by week instead of day

ts_cv <- rolling_origin(dat, initial = 7, assess = 7,
                        skip = 6, cumulative = FALSE)
ts_cv <- tidy(ts_cv)
ggplot(ts_cv, aes(x = Resample, y = factor(Row), fill = Data)) +
  geom_tile() + scale_fill_brewer()
```
two_class_dat  Two Class Data

Details
There are artificial data with two predictors (A and B) and a factor outcome variable (Class).

Value
two_class_dat  a data frame

Examples
data(two_class_dat)
str(two_class_dat)

vfold_cv  V-Fold Cross-Validation

Description
V-fold cross-validation randomly splits the data into V groups of roughly equal size (called "folds"). A resample of the analysis data consisted of V-1 of the folds while the assessment set contains the final fold. In basic V-fold cross-validation (i.e. no repeats), the number of resamples is equal to V.

Usage
vfold_cv(data, v = 10, repeats = 1, strata = NULL, breaks = 4, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A data frame.</td>
</tr>
<tr>
<td>v</td>
<td>The number of partitions of the data set.</td>
</tr>
<tr>
<td>repeats</td>
<td>The number of times to repeat the V-fold partitioning.</td>
</tr>
<tr>
<td>strata</td>
<td>A variable that is used to conduct stratified sampling to create the folds.</td>
</tr>
<tr>
<td>breaks</td>
<td>A single number giving the number of bins desired to stratify a numeric strati-</td>
</tr>
<tr>
<td>...</td>
<td>Not currently used.</td>
</tr>
</tbody>
</table>
Details

The strata argument causes the random sampling to be conducted within the stratification variable. The can help ensure that the number of data points in the analysis data is equivalent to the proportions in the original data set. When more than one repeat is requested, the basic V-fold cross-validation is conducted each time. For example, if three repeats are used with \( v = 10 \), there are a total of 30 splits which as three groups of 10 that are generated separately.

Value

A tibble with classes vfold_cv, rset, tbl_df, tbl, and data.frame. The results include a column for the data split objects and one or more identification variables. For a single repeats, there will be one column called id that has a character string with the fold identifier. For repeats, id is the repeat number and an additional column called idR that contains the fold information (within repeat).

Examples

```r
tibble::tibble(id = LETTERS[1:30])
```
Index

*Topic datasets
  attrition, 5
drinks, 8
two_class_dat, 25

add_resample_id, 2
all.vars(), 8
analysis (as.data.frame.rsplit), 4
apparent, 3
as.data.frame.rsplit, 4
assessment (as.data.frame.rsplit), 4
attrition, 5

bootstraps, 5
bootstraps(), 22
caret2rsample (rsample2caret), 23
complement, 7
drinks, 8
form_pred, 8
gather.rset, 9
group_vfold_cv, 10

initial_split, 11
initial_time_split (initial_split), 11
int_bca (int_pctl), 12
int_pctl, 12
int_t (int_pctl), 12

labels.rset, 14
labels.rsplit, 14
labels.vfold_cv (labels.rset), 14
loo_cv, 15

make_strata, 16
mc_cv, 17
mc_cv(), 22

nested_cv, 19
nested_cv(), 22

populate, 20
populate(), 7

rolling_origin, 21
rolling_origin(), 22
rsample, 22
rsample-package (rsample), 22
rsample2caret, 23

stats::terms(), 8
testing (initial_split), 11
tidy.nested_cv (tidy.rsplit), 23
tidy.rset (tidy.rsplit), 23
tidy.rsplit, 23
tidy.vfold_cv (tidy.rsplit), 23
training (initial_split), 11
two_class_dat, 25

vfold_cv, 25
vfold_cv(), 22