Package 'stablelearner'

May 9, 2026

Version 0.1-7

Date 2025-10-27

Title Stability Assessment of Statistical Learning Methods

Description Graphical and computational methods that can be used to assess the stability of results from supervised statistical learning.

Depends R (>= 3.0.0)

Imports graphics, methods, MASS, e1071, partykit, party, randomForest, ranger

Suggests utils, Formula, nnet, rpart, evtree, rchallenge, knitr, rmarkdown

VignetteBuilder knitr

License GPL-2 | GPL-3

Encoding UTF-8

NeedsCompilation no

Author Michel Philipp [aut],
Carolin Strobl [aut],
Achim Zeileis [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0918-3766>),
Thomas Rusch [aut],
Kurt Hornik [aut] (ORCID: <https://orcid.org/0000-0003-4198-9911>),
Lennart Schneider [aut] (ORCID:
<https://orcid.org/0000-0003-4152-5308>)

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN

Date/Publication 2025-10-27 22:50:02 UTC

Contents

`accuracy` *Prediction Accuracy from Stability Assessment Results*

Description

Function to compute the prediction accuracy from an object of class `"stablelearner"` or `"stablelearnerList"` as a parallel to the similarity values estimated by `stability` in each iteration of the stability assessment procedure.

Usage

```
accuracy(x, measure = "kappa", na.action = na.exclude,
 applyfun = NULL, cores = NULL)
```

Arguments

`x`	an object of class `"stablelearner"` or `"stablelearnerList"`.
`measure`	a character string (or a vector of character strings). Name(s) of the measure(s) used to compute accuracy. Currently implemented measures are `"diag"` = percentage of observations on the main diagonal of a confusion matrix, `"kappa"` = `"diag"` corrected for agreement by chance (default), `"rand"` = Rand index, and `"crand"` = Rand index corrected for agreemend by chance (see also `classAgreement`).
`na.action`	a function which indicates what should happen to the predictions of the results containing `NA`s. The default function is `na.exclude`.
`applyfun`	a `lapply`-like function. The default is to use `lapply` unless `cores` is specified in which case `mclapply` is used (for multicore computations on platforms that support these).
`cores`	integer. The number of cores to use in multicore computations using `mclapply` (see above).

Details

This function can be used to compute prediction accuracy after the stability was estimated using `stability`.

Value

A matrix of size `2*B` times length(`measure`) containing prediction accuracy values of the learners trained during the stability assessment procedure.

See Also

`stability`

Examples

```
library("partykit")
res <- ctree(Species ~ ., data = iris)
stab <- stability(res)
accuracy(stab)
```

---

addLearner *Add Learners to* `LearnerList`

Description

The function can be used to add new learner to `LearnerList` in the current R session.

Usage

```
addLearner(x)
```

Arguments

x	a list containing all required information to define a new learner (see Details below).

Details

The function can be used to add new learners to `LearnerList` in the current R session. The function expects a list of four elements including the name of the learners object class, the name of the package where the class and the fitting method is implemented, the name of the method and a prediction function that predicts class probabilities (in the classification case) or numeric values (in the regression case) and takes the arguments `x` (the fitted model object), `newdata` a `data.frame` containing the predictions of the observations in the evaluation sample and `yclass` a character string specifying the type of the response variable (`"numeric"`, `"factor"`, etc.). The elements in the list should be named `class`, `package`, `method` and `predfun`.

See Also

LearnerList

Examples

```
newlearner <- list(
  class   = "svm",
  package = "e1071",
  method  = "Support Vector Machine",
  predict = function(x, newdata, yclass = NULL) {
    if(match(yclass, c("ordered", "factor"))) {
      attr(predict(x, newdata = newdata, probability = TRUE), "probabilities")
    } else {
      predict(x, newdata = newdata)
    }
  })

addLearner(newlearner)
```

---

bootstrap *Sampler Infrastructure for Stability Assessment*

Description

Sampler objects that provide objects with functionality used by stabletree to generate resampled datasets.

Usage

```
bootstrap(B = 500, v = 1)
subsampling(B = 500, v = 0.632)
samplesplitting(k = 5)
jackknife(d = 1, maxrep = 5000)
splithalf(B = 500)
```

Arguments

B	An integer value specifying the number of resampled datasets.
k	An integer value specifying the number of folds in sample-splitting.
d	An integer value specifying the number of observations left out in jackknife.
maxrep	An integer value specifying the maximum number of resampled datasets allowed, when using jackknife.
v	A numeric value between 0 and 1 specifying the fraction of observations in each subsample.

Details

The sampler functions provide objects that include functionality to generate resampled datasets used by stabletree.

The bootstrap function provides an object that can be used to generate B bootstrap samples by sampling from n observations with replacement.

The subsampling function provides an object that can be used to generate B subsamples by sampling from floor(v*n) observations without replacement.

The samplesplitting function provides an object that can be used to generate k-folds from n observations.

The jackknife function provides an object that can be used to generate all datasets necessary to perform leave-k-out jackknife sampling from n observations. The number of datasets is limited by maxrep to prevent unintended CPU or memory overload by accidently choosing too large values for k.

The splithalf function provides an object that can be used to generate B subsamples by sampling from floor(0.5*n) observations without replacement. When used to implement the "splithalf" resampling strategy for measuring the stability of a result via the stability function, the matrix containing the complement learning samples is generated automatically by stability.

See Also

stabletree, stability

Examples

```
set.seed(0)

## bootstrap sampler
s <- bootstrap(3)
s$sampler(10)

## subsampling
s <- subsampling(3, v = 0.6)
s$sampler(10)

## 5-fold sample-splitting
s <- samplesplitting(5)
s$sampler(10)

## jackknife
s <- jackknife(d = 1)
s$sampler(10)

## splithaf
s <- splithalf(3)
s$sampler(10)
```

`boxplot.stablelearnerList` *Illustrate Results from Stability Assessment*

Description

Illustrates the results from stability assessments performed by `stability` using boxplots.

Usage

```
  ## S3 method for class 'stablelearnerList'
boxplot(x, ..., main = NULL, xlab = NULL, ylab = NULL, reverse = TRUE)
  ## S3 method for class 'stablelearner'
boxplot(x, ...)
```

Arguments

`x`	an object of class `"stablelearnerList"` to be illustrated.
`...`	Arguments passed to `boxplot`.
`main`	a character specifying the title. By default set to `NULL`.
`xlab`	a character specifying the title for the x axis. By default set to `"Learner"`.
`ylab`	a character specifying the title for the y axis. By default set to `NULL`.
`reverse`	logical. If `reverse = TRUE` (default), the similarity values are transformed (reversed) such that higher values indicate a higher stability.

See Also

`stability`, `summary.stablelearnerList`

Examples

```
library("partykit")
r1 <- ctree(Species ~ ., data = iris)

library("rpart")
r2 <- rpart(Species ~ ., data = iris)

stab <- stability(r1, r2, names = c("ctree", "rpart"))
boxplot(stab)
```

dgp_twoclass *Data-Ggnerating Function for Two-Class Problem*

Description

Data-generating function to generate artificial data sets of a classification problem with two response classes, denoted as "A" and "B".

Usage

```
dgp_twoclass(n = 100, p = 4, noise = 16, rho = 0,
  b0 = 0, b = rep(1, p), fx = identity)
```

Arguments

n	integer. Number of observations. The default is 100.
p	integer. Number of signal predictors. The default is 4.
noise	integer. Number of noise predictors. The default is 16.
rho	numeric value between -1 and 1 specifying the correlation between the signal predictors. The correlation is given by rho^k, where k is an integer value given by toeplitz structure. The default is 0 (no correlation between predictors).
b0	numeric value. Baseline probability for class "B" on the logit scale. The default is 0.
b	numeric value. Slope parameter for the predictors on the logit scale. The default is 1 for all predictors.
fx	a function that is used to transform the predictors. The default is identity (equivalent to no transformation).

Value

A data.frame including a column denoted as class that is a factor with two levels "A" and "B". All other columns represent the predictor variables (signal predictors followed by noise predictors) and are named by "x1", "x2", etc..

See Also

stability

Examples

```
dgp_twoclass(n = 200, p = 6, noise = 4)
```

getLearner *Get Learner Details from* LearnerList

Description

Function to get information available about a specific learner in LearnerList of the current R session.

Usage

```
getLearner(x)
```

Arguments

x	a fitted model object.

Details

The function returns the entry in LearnerList found for the class of the object submitted to the function.

See Also

LearnerList, addLearner

Examples

```
library("partykit")
m <- ctree(Species ~ ., data = iris)
getLearner(m)
```

LearnerList *List of Predefined Learners for Assessing Stability*

Description

The list contains details about several predefined learners that are required to assess the stability of results from statistical learning.

Usage

```
LearnerList
```

Details

Currently implemented learners are:

ctree conditional inference trees using `ctree` from **partykit**.

rpart recursive partitioning using `rpart` from **rpart**.

J48 recursive partitioning using `J48` from **RWeka**.

C5.0 recursive partitioning using `C5.0` from **C50**.

tree recursive partitioning using `tree` from **tree**.

lda linear discriminant analysis using `lda` from **MASS**.

lm linear models using `lm` from **stats**.

glm generalized linear models using `glm` from **stats**.

Users can add new learners to `LearnerList` for the current R session, see `addLearner`.

See Also

`addLearner`

---

plot.stabletree *Visualizing Tree Stability Assessments*

Description

Visualizations of tree stability assessments carried out via `stabletree`.

Usage

```
## S3 method for class 'stabletree'
plot(x, select = order(colMeans(x$vs), decreasing = TRUE),
  type.breaks = "levels", col.breaks = "red", lty.breaks = "dashed",
  cex.breaks = 0.7,  col.main = c("black", "gray50"), main.uline = TRUE,
  args.numeric = NULL, args.factor = NULL, args.ordered = NULL, main = NULL,
  original = TRUE, ...)

## S3 method for class 'stabletree'
barplot(height, main = "Variable selection frequencies",
  xlab = "", ylab = "", horiz = FALSE, col = gray.colors(2),
  names.arg = NULL, names.uline = TRUE, names.diag = TRUE,
  cex.names = 0.9,
  ylim = if (horiz) NULL else c(0, 100), xlim = if (horiz) c(0, 100) else NULL,
  original = TRUE, ...)

## S3 method for class 'stabletree'
image(x, main = "Variable selections",
  ylab = "Repetitions", xlab = "", col = gray.colors(2),
```

```
names.arg = NULL, names.uline = TRUE, names.diag = TRUE,
cex.names = 0.9, xaxs = "i", yaxs = "i",
col.tree = 2, lty.tree = 2, xlim = c(0, length(x$vs0)), ylim = c(0, x$B),
original = TRUE, ...)
```

Arguments

x, height	an object of class stabletree.
original	logical. Should the original tree information be highlighted?
select	An vector of integer or character values representing the number(s) or the name(s) of the variable(s) to be plotted. By default all variables are plotted. The numbers correspond to the ordering of all partitioning variables used in the call of the fitted model object that was passed to stabletree.
type.breaks	A character specifying the type of information added to the lines that represent the splits in the complete data tree. type.breaks = "levels" adds the level of the splits. type.breaks = "nodeids" adds the nodeid of the splits. type.breaks = "breaks" adds the cutpoint of the splits. type.breaks = "none" suppresses any labeling.
col.breaks	Coloring of the lines and the texts that represent the splits in the complete data tree.
lty.breaks	Type of the lines that represent the splits in the complete data tree.
cex.breaks	Size of the texts that represent the splits in the complete data tree.
col.main	A vector of colors of length two. The first color is used for titles of variables that are selected in the complete data tree. The second color is used for titles of variables that are not selected in the complete data tree.
main.uline	A logical value. If TRUE, variables selected in the complete data tree are underlined. If FALSE, underlining is suppressed.
args.numeric	A list of arguments passed to the internal function that is used for plotting a histogram of the cutpoints in numerical splits. breaks is passed to hist (see hist for details). Further arguments in the list are passed to plot.histogram such as col, border etc.
args.factor	A list of arguments passed to the internal function that is used for plotting an image plot of the cutpoints in categorical splits. col (a vector of two colors) is used to illustrate categorical splits. Please note that the default colors are optimized for color vision deficiency. col.na defines the color used to highlight missing categories. Further arguments in the list are passed to the function plot.default.
args.ordered	A list of arguments passed to the internal function that is used for plotting a barplot of the cutpoints in ordered categorical splits. All arguments in the list are passed to the function barplot.default, such as col, border etc.
...	further arguments passed to plotting functions, especially for labeling and annotation.
main, xlab, ylab	character. Annotations of axes and main title, respectively.
horiz	A logical value. If FALSE, the bars are drawn vertically with the first bar to the left. If TRUE, bars are drawn horizontally with the first at the bottom.

col	A vector of colors of length two used for coloring in the barplot and image methods. The first color represents selected and the second color represents not selected partitioning variables.
names.arg	A vector of labels to be plotted below each bar (in case of barplot) or each column (in case of image). If the argument is omitted, then the variable names are taken as labels.
names.uline	A logical value. If TRUE, the labels representing variables that are used for splitting in the complete data tree, are underlined. If FALSE, labels are not underlined.
names.diag	A logical value (omitted if horiz = TRUE). If TRUE, the labels are drawn diagonally. If FALSE, labels are drawn horizontally.
cex.names	Expansion factor for labels.
xlim, ylim	The limits of the plot.
xaxs, yaxs	The style of axis interval calculation to be used (see par for details).
col.tree, lty.tree	color and line type to indicate differences from the original tree that was resampled.

Details

- plot visualizes the variability of the cutpoints.
- barplot visualizes the variable selection frequency.
- image visualizes the combinations of variables selected.

See Also

stabletree

Examples

```
## build a tree
library("partykit")
m <- ctree(Species ~ ., data = iris)
plot(m)

## investigate stability
set.seed(0)
s <- stabletree(m, B = 500)

## show variable selection proportions
## with different labels and different ordering
barplot(s)
barplot(s, cex.names = 0.8)
barplot(s, names.diag = FALSE)
barplot(s, names.arg = c("a", "b", "c", "d"))
barplot(s, names.uline = FALSE)
barplot(s, col = c("lightgreen", "darkred"))
```

```
barplot(s, horiz = TRUE)

## illustrate variable selections of replications
## with different labels and different ordering
image(s)
image(s, cex.names = 0.8)
image(s, names.diag = FALSE)
image(s, names.arg = c("a", "b", "c", "d"))
image(s, names.uline = FALSE)
image(s, col = c("lightgreen", "darkred"))

## graphical cutpoint analysis, selecting variable by number and name
## with different numerical of break points
plot(s)
plot(s, select = 3)
plot(s, select = "Petal.Width")
plot(s, args.numeric = list(breaks = 40))

# change labels of splits in complete data tree
plot(s, select = 3, type.breaks = "levels")
plot(s, select = 3, type.breaks = "nodeids")
plot(s, select = 3, type.breaks = "breaks")
plot(s, select = 3, type.breaks = "none")
```

`similarity_measures_classification` *Similarity Measure Infrastructure for Stability Assessment with Ordinal Responses*

Description

Functions that provide objects with functionality used by `stability` to measure the similarity between the predictions of two results in classification problems.

Usage

```
clagree()
ckappa()

bdist()
tvdist()
hdist()
jsdiv(base = 2)
```

Arguments

base	A positive or complex number: the base with respect to which logarithms are computed. Defaults to 2.

Details

The similarity measure functions provide objects that include functionality used by stability to measure the similarity between the probability predictions of two results in classification problems.

The clagree and ckappa functions provide an object that can be used to assess the similarity based on the predicted classes of two results. The predicted classes are selected by the class with the highest probability.

The bdist (Bhattacharayya distance), tvdist (Total variation distance), hdist (Hellinger distance) and jsdist (Jenson-Shannon divergence) functions provide an object that can be used to assess the similarity based on the predicted class probabilities of two results.

See Also

stability

Examples

```
set.seed(0)

## build trees
library("partykit")
m1 <- ctree(Species ~ ., data = iris[sample(1:nrow(iris), replace = TRUE),])
m2 <- ctree(Species ~ ., data = iris[sample(1:nrow(iris), replace = TRUE),])

p1 <- predict(m1, type = "prob")
p2 <- predict(m2, type = "prob")

## class agreement
m <- clagree()
m$measure(p1, p2)

## cohen's kappa
m <- ckappa()
m$measure(p1, p2)

## bhattacharayya distance
m <- bdist()
m$measure(p1, p2)

## total variation distance
m <- tvdist()
m$measure(p1, p2)

## hellinger distance
```

```
m <- hdist()
m$measure(p1, p2)

## jenson-shannon divergence
m <- jsdiv()
m$measure(p1, p2)

## jenson-shannon divergence (base = exp(1))
m <- jsdiv(base = exp(1))
m$measure(p1, p2)
```

`similarity_measures_regression` *Similarity Measure Infrastructure for Stability Assessment with Numerical Responses*

Description

Functions that provide objects with functionality used by `stability` to measure the similarity between numeric predictions of two results in regression problems.

Usage

```
edist()
msdist()
rmsdist()
madist()
qadist(p = 0.95)

cprob(kappa = 0.1)
rbfkernel()
tanimoto()
cosine()

ccc()
pcc()
```

Arguments

`p`	A numeric value between 0 and 1 specifying the probability to which the sample quantile of the absolute distance between the predictions is computed.
`kappa`	A positive numeric value specifying the upper limit of the absolute distance between the predictions to which the coverage probability is computed.

Details

The similarity measure functions provide objects that include functionality used by `stability` to measure the similarity between numeric predictions of two results in regression problems.

The `edist` (euclidean distance), `msdist` (mean squared distance), `rmsdist` (root mean squared distance), `madist` (mean absolute distance) and `qadist` (quantile of absolute distance) functions implement scale-variant distance measures that are unbounded.

The `cprob` (coverage probability), `rbfkernel` (gaussian radial basis function kernel), `tanimoto` (tanimoto coefficient) and `cosine` (cosine similarity) functions implement scale-variant distance measures that are bounded.

The `ccc` (concordance correlation coefficient) and `pcc` (pearson correlation coefficient) functions implement scale-invariant distance measures that are bounded between 0 and 1.

See Also

`stability`

Examples

```
set.seed(0)

library("partykit")

airq <- subset(airquality, !is.na(Ozone))
m1 <- ctree(Ozone ~ ., data = airq[sample(1:nrow(airq), replace = TRUE),])
m2 <- ctree(Ozone ~ ., data = airq[sample(1:nrow(airq), replace = TRUE),])

p1 <- predict(m1)
p2 <- predict(m2)

## euclidean distance
m <- edist()
m$measure(p1, p2)

## mean squared distance
m <- msdist()
m$measure(p1, p2)

## root mean squared distance
m <- rmsdist()
m$measure(p1, p2)

## mean absolute istance
m <- madist()
m$measure(p1, p2)

## quantile of absolute distance
m <- qadist()
m$measure(p1, p2)
```

```
## coverage probability
m <- cprob()
m$measure(p1, p2)

## gaussian radial basis function kernel
m <- rbfkernel()
m$measure(p1, p2)

## tanimoto coefficient
m <- tanimoto()
m$measure(p1, p2)

## cosine similarity
m <- cosine()
m$measure(p1, p2)

## concordance correlation coefficient
m <- ccc()
m$measure(p1, p2)

## pearson correlation coefficient
m <- pcc()
m$measure(p1, p2)
```

`similarity_values` *Extracting Similarity Values*

Description

Extract similarity values from object returned by `stability` for further illustration or analysis.

Usage

```
similarity_values(x, reverse = TRUE)
```

Arguments

`x`	an object of class `"stablelearner"` or `"stablelearnerList"` from which the similarity values are extracted.
`reverse`	logical. If `reverse = TRUE` (default), the similarity values are transformed (reversed) such that higher values indicate a higher stability.

Value

A numeric array of dimension 3 containing similarity values. The dimensions represent repetitions, results (fitted model objects) and similarity measures.

See Also

stability, summary.stablelearnerList

Examples

```
library("partykit")
res <- ctree(Species ~ ., data = iris)
stab <- stability(res)
similarity_values(stab)
```

---

stability *Stability Assessment for Results from Supervised Statistical Learning*

Description

Stability assessment of results from supervised statistical learning (i.e., recursive partitioning, support vector machines, neural networks, etc.). The procedure involves the pairwise comparison of results generated from learning samples randomly drawn from the original data set or directly from the data-generating process (if available).

Usage

```
stability(x, ..., data = NULL, control = stab_control(), weights = NULL,
  applyfun = NULL, cores = NULL, names = NULL)
```

Arguments

x	fitted model object. Any model object can be used whose class is registered in LearnerList. Users can add classes for the current R session to LearnerList, see addLearner.
...	additional fitted model objects.
data	an optional data.frame or a data-generating function. By default the learning data from x is used (if this can be inferred from the getCall of x.
control	a list with control parameters, see stab_control.
weights	an optional matrix of dimension n * B that can be used to weight the observations from the original learning data when the models are refitted. If weights = true, the weights are computed internally according to the sampler defined in control. If weight = NULL (default), no case-weights are used and the sampler defined in control will be applied to the original data set.
applyfun	a lapply-like function. The default is to use lapply unless cores is specified in which case mclapply is used (for multicore computations on platforms that support these).
cores	integer. The number of cores to use in multicore computations using mclapply (see above).
names	a vector of characters to specify a name for each fitted model object. By default, the objects are named by their class.

Details

Assesses the (overall) stability of a result from supervised statistical learning by quantifying the similarity of realizations from the distribution of possible results (given the algorithm, the formulated model, the data-generating process, the sample size, etc.). The stability distribution is estimated by repeatedly assessing the similarity between the results generated by training the algorithm on two different learning samples, by means of a similarity metric. The learning samples are generated by sampling from the learning data or the data-generating process in case of a simulation study. For more details, see Philipp et al. (2018).

Value

For a single fitted model object, `stability` returns an object of class `"stablelearner"` with the following components:

`call`	the call from the model object `x`,
`learner`	the information about the learner retrieved from `LearnerList`,
`B`	the number of repetitions,
`sval`	a matrix containing the estimated similarity values for each similarity measure specified in `control`,
`sampstat`	a list containing information on the size of the learning samples (`ls`), the size of the overlap between the learning samples (`lo`), the size of the evaluation sample (`es`) and the size of the overlap between the evaluation and the learning samples (`eo`) in each repetition.
`data`	a language object referring to the `data.frame` or the data-generating function used for assessing the stability,
`control`	a list with control parameters used for assessing the stability,

For several fitted model objects, `stability` returns an object of class `"stablelearnerList"` which is a list of objects of class `"stablelearner"`.

References

Philipp M, Rusch T, Hornik K, Strobl C (2018). "Measuring the Stability of Results from Supervised Statistical Learning". *Journal of Computational and Graphical Statistics*, **27**(4), 685–700. doi:10.1080/10618600.2018.1473779

See Also

`boxplot.stablelearnerList`, `summary.stablelearner`

Examples

```
## assessing the stability of a single result
library("partykit")
r1 <- ctree(Species ~ ., data = iris)
stab <- stability(r1)
summary(stab)
```

```
## assessing the stability of several results
library("rpart")
r2 <- rpart(Species ~ ., data = iris)
stab <- stability(r1, r2, control = stab_control(seed = 0))
summary(stab, names = c("ctree", "rpart"))

## using case-weights instead of resampling
stability(r1, weights = TRUE)

## using self-defined case-weights
n <- nrow(iris)
B <- 500
w <- array(sample(c(0, 1), size = n * B * 3, replace = TRUE), dim = c(n, B, 3))
stability(r1, weights = w)

## assessing stability for a given data-generating process
my_dgp <- function() dgp_twoclass(n = 100, p = 2, noise = 4, rho = 0.2)
res <- ctree(class ~ ., data = my_dgp())
stability(res, data = my_dgp)
```

`stabletree` *Stability Assessment for Tree Learners*

Description

Stability assessment of variable and cutpoint selection in tree learners (i.e., recursive partitioning). By refitting trees to resampled versions of the learning data, the stability of the trees structure is assessed and can be summarized and visualized.

Usage

```
stabletree(x, data = NULL, sampler = subsampling, weights = NULL,
  applyfun = NULL, cores = NULL, savetrees = FALSE, ...)
```

Arguments

`x`	fitted model object. Any tree-based model object that can be coerced by `as.party` can be used provided that suitable methods are provided.
`data`	an optional `data.frame`. By default the learning data from `x` is used (if this can be inferred from the `getCall` of `x`.
`sampler`	a resampling (generating) function. Either this should be a function of `n` that returns a matrix or a sampler generator like `subsampling`.
`weights`	an optional matrix of dimension n * B that can be used to weight the observations from the original learning data when the trees are refitted. If `weight = NULL`, the `sampler` will be used.

applyfun	a lapply-like function. The default is to use lapply unless cores is specified in which case mclapply is used (for multicore computations on platforms that support these).
cores	integer. The number of cores to use in multicore computations using mclapply (see above).
savetrees	logical. If TRUE, trees based on resampled data sets are returned.
...	further arguments passed to sampler.

Details

The function stabletree assesses the stability of tree learners (i.e., recursive partitioning methods) by refitting the tree to resampled versions of the learning data. By default, if data = NULL, the fitting call is extracted by getCall to infer the learning data. Subsequently, the sampler generates B resampled versions of the learning data, the tree is regrown with update, and (if necessary) coerced by as.party. For each of the resampled trees it is queried and stored which variables are selected for splitting and what the selected cutpoints are.

The resulting object of class "stabletree" comes with a set of standard methods to generic functions including print, summary for numerical summaries and plot, barplot, and image for graphical representations. See plot.stabletree for more details. In most methods, the argument original can be set to TRUE or FALSE, turning highlighting of the original tree information on and off.

Value

stabletree returns an object of class "stabletree" which is a list with the following components:

call	the call from the model object x,
B	the number of resampled datasets,
sampler	the sampler function,
vs0	numeric vector of the variable selections of the original tree,
br0	list of the break points (list of nodeids, levels, and breaks) for each variable of the original tree,
vs	numeric matrix of the variable selections for each resampled dataset,
br	list of the break points (only the breaks for each variable over all resampled datasets,
classes	character vector indicating the classes of all partitioning variables,
trees	a list of tree objects of class "party", or NULL.

References

Hothorn T, Zeileis A (2015). partykit: A Modular Toolkit for Recursive Partytioning in R. *Journal of Machine Learning Research*, **16**(118), 3905–3909.

Philipp M, Zeileis A, Strobl C (2016). "A Toolkit for Stability Assessment of Tree-Based Learners". In A. Colubi, A. Blanco, and C. Gatu (Eds.), Proceedings of COMPSTAT 2016 – 22nd International Conference on Computational Statistics (pp. 315–325). The International Statistical Institute/International Association for Statistical Computing. Preprint available at https://EconPapers.RePEc.org/RePEc:inn:wpaper:2016-11

See Also

plot.stabletree, as.stabletree, as.party

Examples

```
## build a simple tree
library("partykit")
m <- ctree(Species ~ ., data = iris)
plot(m)

## investigate stability
set.seed(0)
s <- stabletree(m, B = 500)
print(s)

## variable selection statistics
summary(s)

## show variable selection proportions
barplot(s)

## illustrate variable selections of replications
image(s)

## graphical cutpoint analysis
plot(s)
```

---

stabletree-coercion *Coercion Functions*

Description

Functions coercing various forest objects to objects of class "stabletree".

Usage

```
  as.stabletree(x, ...)
  ## S3 method for class 'randomForest'
as.stabletree(x, applyfun = NULL, cores = NULL, ...)
  ## S3 method for class 'RandomForest'
as.stabletree(x, applyfun = NULL, cores = NULL, ...)
  ## S3 method for class 'cforest'
as.stabletree(x, applyfun = NULL, cores = NULL, savetrees = FALSE, ...)
  ## S3 method for class 'ranger'
as.stabletree(x, applyfun = NULL, cores = NULL, ...)
```

Arguments

x	an object of class randomForest, RandomForest-class, cforest, or ranger.
applyfun	a lapply-like function. The default is to use lapply unless cores is specified in which case mclapply is used (for multicore computations on platforms that support these).
cores	integer. The number of cores to use in multicore computations using mclapply (see above).
savetrees	logical. If TRUE, trees of the forests are returned.
...	additional arguments (currently not used).

Details

Random forests fitted using randomForest, cforest, cforest or ranger are coerced to "stabletree" objects.

Note that when plotting a randomForest or ranger, the gray areas of levels of a nominal variable do not mimic exactly the same behavior as for classical "stabletree" objects, due to randomForest and ranger, not storing any information whether any individuals were left fulfilling the splitting criterion in the subsample. Therefore, gray areas only indicate that this level of this variable has already been used in a split before in such a way that it could not be used for any further splits.

For ranger, interaction terms are (currently) not supported.

Value

as.stabletree returns an object of class "stabletree" which is a list with the following components:

call	the call from the model object x,
B	the number of trees of the random forest,
sampler	the random forest fitting function,
vs0	numeric vector of the variable selections of the original tree, here always a vector of zeros because there is no original tree,
br0	always NULL (only included for consistency),
vs	numeric matrix of the variable selections for each tree of the random forest,
br	list of the break points (only the breaks for each variable over all trees of the random forest,
classes	character vector indicating the classes of all partitioning variables,
trees	a list of tree objects of class "party", or NULL.

See Also

stabletree, plot.stabletree

Examples

```
## build a randomForest using randomForest
library(randomForest)
set.seed(1)
rf <- randomForest(Species ~ ., data = iris)

## coerce to a stabletree
srf <- as.stabletree(rf)
print(srf)
summary(srf, original = FALSE) # there is no original tree
barplot(srf)
image(srf)
plot(srf)

## build a RandomForest using party
library("party")
set.seed(2)
cf_party <- cforest(Species ~ ., data = iris,
  control = cforest_unbiased(mtry = 2))

## coerce to a stabletree
scf_party <- as.stabletree(cf_party)
print(scf_party)
summary(scf_party, original = FALSE)
barplot(scf_party)
image(scf_party)
plot(scf_party)

## build a cforest using partykit
library("partykit")
set.seed(3)
cf_partykit <- cforest(Species ~ ., data = iris)

## coerce to a stabletree
scf_partykit <- as.stabletree(cf_partykit)
print(scf_partykit)
summary(scf_partykit, original = FALSE)
barplot(scf_partykit)
image(scf_partykit)
plot(scf_partykit)

## build a random forest using ranger
library("ranger")
set.seed(4)
rf_ranger <- ranger(Species ~ ., data = iris)

## coerce to a stabletree
srf_ranger <- as.stabletree(rf_ranger)
print(srf_ranger)
summary(srf_ranger, original = FALSE)
barplot(srf_ranger)
```

```
image(srf_ranger)
plot(srf_ranger)
```

---

stab_control *Control for Supervised Stability Assessments*

Description

Various parameters that control aspects of the stability assessment performed via stability.

Usage

```
stab_control(B = 500, measure = list(tvdist, ccc), sampler = "bootstrap",
  evaluate = "OOB", holdout = 0.25, seed = NULL, na.action = na.exclude,
  savepred = TRUE, silent = TRUE, ...)
```

Arguments

B	an integer value specifying the number of repetitions. The default is B = 500.
measure	a list of similarity measure (generating) functions. Those should either be functions of p1 and p2 (the predictions of two results in a repetition, see Details below) or similarity measure generator functions, see similarity_measures_classification and similarity_measures_regression. The defaults are tvdist for the classification and ccc for the regression case.
sampler	a resampling (generating) function. Either this should be a function of n that returns a matrix or a sampler generator like bootstrap. The default is "bootstrap".
evaluate	a character specifying the evaluation strategy to be applied (see Details below). The default is "OOB".
holdout	a numeric value between zero and one that specifies the proportion of observations hold out for evaluation over all repetitions, only if evaluate = "OOB". The default is 0.25.
seed	a single value, interpreted as an integer, see set.seed and Details section below. The default is NULL, which indicates that no seed has been set.
na.action	a function which indicates what should happen when the predictions of the results contain NAs. The default function is na.exclude.
savepred	logical. Should the predictions from each iteration be saved? If savepred = FALSE, the resulting object will be smaller. However, predictions are required to subsequently compute measures of accuracy via accuracy.
silent	logical. If TRUE, error messages, generated by the learner while training or predicting, are suppressed and NA is returned for the corresponding iteration of the stability assessment procedure.
...	arguments passed to sampler function.

Details

With the argument `measure` one or more measures can be defined that are used to assess the stability of a result from supervised statistical learning by `stability`. Predefined similarity measures for the regression and the classification case are listed in `similarity_measures_classification` and `similarity_measures_regression`.

Users can define their own similarity functions `f(p1, p2)` that must return a single numeric value for the similarity between two results trained on resampled data sets. Such a function must take the arguments `p1` and `p2`. In the classification case, `p1` and `p2` are probability matrices of size m * K, where `m` is the number of predicted observations (size of the evaluation sample) and K is the number of classes. In the regression case, `p1` and `p2` are numeric vectors of length m.

A different way to implement new similarity functions for the current R session is to define a similarity measure generator function, which is a function without arguments that generates a list of five elements including the name of the similarity measure, the function to compute the similarity between the predictions as described above, a vector of character values specifying the response types for which the similarity measure can be used, a list containing two numeric elements `lower` and `upper` that specify the range of values of the similarity measure and the function to invert (or reverse) the similarity values such that higher values indicate higher stability. The latter can be set to `NULL`, if higher similarity values already indicate higher stability. Those elements should be named `name`, `measure`, `classes`, `range` and `reverse`.

The argument `evaluate` can be used to specify the evaluation strategy. If set to `"ALL"`, all observations in the original data set are used for evaluation. If set to `"OOB"`, only the pairwise out-of-bag observations are used for evaluation within each repetition. If set to `"OOS"`, a fraction (defined by `holdout`) of the observations in the original data set are randomly sampled and used for evaluation, but not for training, over all repetitions.

The argument `seed` can be used to make similarity assessments comparable when comparing the stability of different results that were trained on the same data set. By default, `seed` is set to `NULL` and the learning samples are sampled independently for each fitted model object passed to `stability`. If `seed` is set to a specific number, the seed will be set for each fitted model object before the learning samples are generated using `"L'Ecuyer-CMRG"` (see `set.seed`) which guarantees identical learning samples for each stability assessment and, thus, comparability of the stability assessments between the results.

See Also

`stability`

Examples

```
library("partykit")
res <- ctree(Species ~ ., data = iris)

## less repetitions
stability(res, control = stab_control(B = 100))
```

```
## Not run:

## change similarity measure
stability(res, control = stab_control(measure = list(bdist)))

## change evaluation strategy
stability(res, control = stab_control(evaluate = "ALL"))
stability(res, control = stab_control(evaluate = "OOS"))

## change resampling strategy to subsampling
stability(res, control = stab_control(sampler = subsampling))
stability(res, control = stab_control(sampler = subsampling, evaluate = "ALL"))
stability(res, control = stab_control(sampler = subsampling, evaluate = "OOS"))

## change resampling strategy to splithalf
stability(res, control = stab_control(sampler = splithalf, evaluate = "ALL"))
stability(res, control = stab_control(sampler = splithalf, evaluate = "OOS"))


## End(Not run)
```

`summary.stablelearnerList`
Summarize Results from Stability Assessment

Description

Summarizes and prints the results from stability assessments performed by `stability`.

Usage

```
## S3 method for class 'stablelearnerList'
summary(object, ..., reverse = TRUE,
  probs = c(0.05, 0.25, 0.5, 0.75, 0.95), digits = 3, names = NULL)
```

Arguments

`object`	a object of class `"stablelearner"` or `"stablelearnerList"` to be summarized.
`...`	Arguments passed from or to other functions (currently ignored).
`reverse`	logical. If `reverse = TRUE` (default), the similarity values summarized in the `summary` output are transformed (reversed) such that higher values indicate a higher stability.
`digits`	integer. The number of digits used to summarize the similarity distribution in the `summary` output.
`probs`	a vector of probabilities used tosummarize the similarity distribution in the `summary` output.

names a vector of characters to specify a name for each result from statistical learning in the summary output.

See Also

stability

Examples

```
library("partykit")

rval <- ctree(Species ~ ., data = iris)
stab <- stability(rval)

summary(stab)
summary(stab, reverse = FALSE)
summary(stab, probs = c(0.25, 0.5, 0.75))
summary(stab, names = "conditional inference tree")
```

---

titanic *Passengers and Crew on the RMS Titanic*

Description

the titanic data is a complete list of passengers and crew members on the RMS Titanic. It includes a variable indicating whether a person did survive the sinking of the RMS Titanic on April 15, 1912.

Usage

```
data("titanic")
```

Format

A data frame containing 2207 observations on 11 variables.

name a string with the name of the passenger.

gender a factor with levels male and female.

age a numeric value with the persons age on the day of the sinking. The age of babies (under 12 months) is given as a fraction of one year (1/month).

class a factor specifying the class for passengers or the type of service aboard for crew members.

embarked a factor with the persons place of of embarkment.

country a factor with the persons home country.

ticketno a numeric value specifying the persons ticket number (NA for crew members).

fare a numeric value with the ticket price (NA for crew members, musicians and employees of the shipyard company).

sibsp an ordered factor specifying the number if siblings/spouses aboard; adopted from Vanderbild data set (see below).

parch an ordered factor specifying the number of parents/children aboard; adopted from Vanderbild data set (see below).

survived a factor with two levels (no and yes) specifying whether the person has survived the sinking.

Details

The website https://www.encyclopedia-titanica.org/ offers detailed information about passengers and crew members on the RMS Titanic. According to the website 1317 passengers and 890 crew member were abord.

8 musicians and 9 employees of the shipyard company are listed as passengers, but travelled with a free ticket, which is why they have NA values in fare. In addition to that, fare is truely missing for a few regular passengers.

Source

The complete list of persons on the RMS titanic was downloaded from https://www.encyclopedia-titanica.org/ on April 5, 2016. The information given in sibsp and parch was adopoted from a data set obtained from https://hbiostat.org/data/.

References

https://www.encyclopedia-titanica.org/ and https://hbiostat.org/data/.

Examples

```
data("titanic", package = "stablelearner")
summary(titanic)
```

---

tuner *Tuning Wrapper Function*

Description

Convenience function to train a method using different tuning parameters.

Usage

```
tuner(method, tunerange, ...)
```

Arguments

method	a character string. Name of the R function to train the method.
tunerange	a list. A list that specifies the range of values to be used for each tuning parameter. Each element of the list should be a vector that specifies the values to be tested for the tuning parameter. The element must be named after the corresponding tuning parameter of the method (see examples).
...	additional information passed to method (such as formula, data, subset, etc.).

Details

This function can be used to train any method using different values for its tuning parameter(s). The result can be passed directly to stability to compare the stability of results based on different values of the tuning parameter.

Value

A list that contains all fitted model objects.

Additional information about the range of values used for the tuning parameters is attached to the resulting object as an attribute.

See Also

stability

Examples

```
library("partykit")

## tuning cforest using different values of its tuning parameter mtry
r <- tuner("cforest", tunerange = list(mtry = 1:4), formula = Species ~ ., data = iris)

## assess stability (with B = 10 for illustration to avoid excessive computation times)
stability(r, control = stab_control(seed = 1234, B = 10))

## receive information about the range of tuning parameters
attr(r, "range")
```

Index