Package ‘treemisc’

October 19, 2022

Type Package

Title Data Sets and Functions to Accompany ``Tree-Based Methods for Statistical Learning in R''

Version 0.0.1

Description Miscellaneous data sets and functions to accompany Greenwell (2022) ``Tree-Based Methods for Statistical Learning in R'' <doi:10.1201/9781003089032>.

License GPL (>= 2)

Depends R (>= 4.0.0)

Imports Matrix, Rcpp (>= 1.0.1), rpart, stats, utils

Suggests AmesHousing, gbm, glmnet (>= 3.0), graphics, mlbench, party, partykit, randomForest, ranger, rpart.plot, tinytest

LinkingTo Rcpp

RoxygenNote 7.2.1

LazyData TRUE

Encoding UTF-8

NeedsCompilation yes

Author Brandon M. Greenwell [aut, cre] (<`https://orcid.org/0000-0002-8120-0084`>)

Maintainer Brandon M. Greenwell <`greenwell.brandon@gmail.com`>

Repository CRAN

Date/Publication 2022-10-19 10:10:02 UTC

R topics documented:

`treemisc-package` *Data Sets and Functions to Accompany "Tree-Based Methods for Statistical Learning in R"*

Description

Miscellaneous data sets and functions to accompany Greenwell (2022) "Tree-Based Methods for Statistical Learning in R" <doi:10.1201/9781003089032>.

Details

The DESCRIPTION file:

Package:	treemisc
Type:	Package
Title:	Data Sets and Functions to Accompany "Tree-Based Methods for Statistical Learning in R"
Version:	0.0.1
Authors@R:	person(c("Brandon", "M."), family = "Greenwell", email = "greenwell.brandon@gmail.com", role = c("aut",
Description:	Miscellaneous data sets and functions to accompany Greenwell (2022) "Tree-Based Methods for Statistical I
License:	GPL (>= 2)
Depends:	R (>= 4.0.0)
Imports:	Matrix, Rcpp (>= 1.0.1), rpart, stats, utils
Suggests:	AmesHousing, gbm, glmnet (>= 3.0), graphics, mlbench, party, partykit, randomForest, ranger, rpart.plot, tin
LinkingTo:	Rcpp
RoxygenNote:	7.2.1

LazyData:	TRUE
Encoding:	UTF-8
Author:	Brandon M. Greenwell [aut, cre] (<https://orcid.org/0000-0002-8120-0084>)
Maintainer:	Brandon M. Greenwell <greenwell.brandon@gmail.com>

Index of help topics:

```
banknote                Swiss banknote data
banknote2               Swiss banknote data (UCI version)
calibrate               External probability calibration
cummean                 Cumulative means
decision_boundary       Add decision boundary to a scatterplot.
gbm_2way                Two-way interactions
gen_friedman1           Friedman benchmark data
gen_mease               Generate data from the Mease model
guide_setup             Generate GUIDE input files
hitters                 Baseball data (corrected)
isle_post               Importance sampled learning ensemble
ladboost                Gradient tree boosting with least absolute
                        deviation (LAD) loss
lift                    Gain and lift charts
load_eslmix             Gaussian mixture data
lsboost                 Gradient tree boosting with least squares (LS)
                        loss
mushroom                Mushroom edibility
predict.rforest         Random forest predictions
proximity               Proximity matrix
prune_se                Prune an 'rpart' object
rforest                 Random forest
rrm                     Random rotation matrix
tree_diagram            Tree diagram
treemisc-package        Data Sets and Functions to Accompany
                        "Tree-Based Methods for Statistical Learning in
                        R"
wilson_hilferty         Modified Wilson-Hilferty approximation
wine                    Wine quality
xy_grid                 Create a Cartesian product from evenly spaced
                        values of two variables
```

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

NA

Maintainer: NA

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
## Optional simple examples of the most important functions
## Use \dontrun{} around code to be shown but not executed
```

`banknote` *Swiss banknote data*

Description

Measurements from 200 Swiss 1000-franc banknotes: 100 genuine and 100 counterfeit.

Format

A data frame with 200 rows and 7 columns.

Details

length The length of the bill in mm.

left The length of the left edge in mm.

right The length of the right edge in mm.

bottom The length of the botttom edge in mm.

top The length of the top edge in mm.

diagonal The length of the diagonal in mm.

y Integer specifying whether or not the bill was genuine (`y = 0`) or counterfeit (`y = 1`).

Source

Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics: A practical approach*. London: Chapman & Hall.

`banknote2` *Swiss banknote data (UCI version)*

Description

Data were extracted from images that were taken from genuine `class = 1` and forged `class = 0` banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images contained 400 x 400 pixels. Due to the object lens and distance to the investigated object, gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet transformation tools were used to extract features from the images.

Format

A data frame with 1372 rows and 5 variables.

Details

vow Variance of the wavelet transformed image (continuous)

sow Skewness of the wavelet transformed image (continuous)

kow Kurtosis of the wavelet transformed image (continuous)

eoi Entropy of the image (continuous)

class Integer specifying whether or not the specimen was genuine (`class = 1`) or forged (`class = 0`).

Source

Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

`calibrate` *External probability calibration*

Description

Validates predicted probabilities against a set of observed (binary) outcomes.

Usage

```
calibrate(
  prob,
  y,
  method = c("pratt", "iso", "ns", "bins"),
  pos.class = NULL,
  probs = c(0.05, 0.35, 0.65, 0.95),
  nbins = 10
)

## S3 method for class 'calibrate'
print(x, ...)

## S3 method for class 'calibrate'
plot(
  x,
  refline = TRUE,
  refline.col = 2,
  refline.lty = "dashed",
  refline.lwd = 1,
  ...
)
```

Arguments

`prob`	Vector of predicted probabilities.
`y`	Vector of binary (i.e., 0/1) outcomes. If `y` is coded as anything other than 0/1, then you must specify which of the two categories represents the "positive" class (i.e., the class for which the probabilities specified in `prob` correspond to) via the `pos.class` argument.
`method`	Character string specifying which calibration method to use. Current options include: `"pratt"` Pratt scaling. `"iso"` Isotonic (i.e., monotonic) calibration. `"ns"` Natural (i.e., restricted) cubic splines; essentially, a spline-based nonparametric version of Pratt scaling. `"binned"` TBD.
`pos.class`	Numeric/character string specifying which values in `y` correspond to the "positive" class. Default is `NULL`. (Must be specified whenever `y` is not coded as 0/1., where 1 is assumed to represent the "positive" class.)
`probs`	Numeric vector specifying the probabilities for generating the quantiles of `prob` on the logit scale; these are used for the knot locations defining the spline whenever `method = "ns"`. The default corresponds to a good choice based on four knots; see Harrel (2015, pp. 26-28) for details.
`nbins`	Integer specifying the number of bins to use for grouping the probabilities.
`x`	An object of class `"calibrate"`.
`...`	Additional optional argument to be passed on to other methods.
`refline`	Logical indicating whether or not to include a reference line.
`refline.col`	The color to use for the reference line. Default is `"red"`.
`refline.lty`	The type of line to use for the reference line. Default is `"dashed"`.
`refline.lwd`	The width of the reference line. Default is 1.

Value

A "calibrate" object, which is essentially a list with the following components:

"probs" A data frame containing two columns: original (the original probability estimates) and calibrated (the calibrated probability estimates).

"calibrater" The calibration function (essentially a fitted model object) which can be used to calibrate new probabilities.

"bs" The Brier score between prob and y.

References

Harrell, Frank. (2015). Regression Modeling Strategies. Springer Series in Statistics. Springer International Publishing.

---

cummean *Cumulative means*

Description

Returns a vector whose elements are the cumulative means of the argument.

Usage

```
cummean(x)
```

Arguments

x	A numeric object.

Value

A vector of the same length and type as x (after coercion). Names are preserved.

An NA value in x causes the corresponding and following elements of the return value to be NA, as does integer overflow (with a warning).

Examples

```
x <- 1:10
cummean(x)
cumsum(x) / seq_along(x)  # equivalent using cumulative sums
```

`decision_boundary` *Add decision boundary to a scatterplot.*

Description

Adds the decision boundary from a classification model (binary or multiclass) to an existing scatterplot.

Usage

```
decision_boundary(model, train, y, x1, x2, pfun, grid.resolution = 100, ...)

## Default S3 method:
decision_boundary(
  model,
  train,
  y,
  x1,
  x2,
  pfun = NULL,
  grid.resolution = 100,
  ...
)
```

Arguments

`model`	The associated model object.
`train`	Data frame of training observations.
`y`	Character string giving the name of the outcome variable in `train`.
`x1`	Character string giving the name of the predictor in `train` that corresponds to the x-axis.
`x2`	Character string giving the name of the predictor in `train` that corresponds to the y-axis.
`pfun`	Optional prediction wrapper that returns a vector of predicted class labels. It must have exactly two arguments: `object` and `newdata`.
`grid.resolution`	Integer specifying the resolution of the contour plot. Default is `100`.
`...`	Additional optional arguments to be passed on to `contour`.

Value

No return value, only called for side effects; in this case, a contour displaying the decision boundary of a classifier is added to an existing scatterplot.

Note

Based on a function written by Michael Hahsler; see https://michael.hahsler.net/SMU/EMIS7332/R/viz_classifier.html.

Examples

```
## Not run:
library(mlbench)
library(rpart)
library(treemisc)

# Generate training data from the twonorm benchmark problem
set.seed(1050)  # for reproducibility
trn <- as.data.frame(mlbench.twonorm(500, d = 2))

# Fit a default classification tree
tree <- rpart(classes ~ ., data = trn)

# Scatterplot of training data
palette("Okabe-Ito")
plot(x.2 ~ x.1, data = trn, col = as.integer(trn$classes) + 1,
     xlab = expression(x[1]), ylab = expression(x[2]))
palette("default")

# Add a decision boundary
decision_boundary(tree, train = trn, y = "y", x1 = "x.1", x2 = "x.2")

## End(Not run)
```

`gbm_2way` *Two-way interactions*

Description

Computes Friedman's H-statistic (Friedman & Popescu, 2008) for all pairwise interaction effects in a `gbm` model.

Usage

```
gbm_2way(object, data, var.names = object$var.names, n.trees = object$n.trees)
```

Arguments

`object`	A `gbm` object.
`data`	Data frame containing the original training data (or representative sample thereof).
`var.names`	Character string specifying the predictor names to consider.
`n.trees`	Integer specifying the number of trees to use.

Value

A data frame with the following three columns:

var1 The name of the first feature.

var2 The name of the second feature.

h The corresponding H-statistic.

The resulting rows are sorted in descending order of `h`.

References

Friedman, J. H., & Popescu, B. E. (2008). Predictive Learning via Rule Ensembles. The Annals of Applied Statistics, 2(3), 916–954. http://www.jstor.org/stable/30245114

`gen_friedman1` *Friedman benchmark data*

Description

Simulate data from the Friedman 1 benchmark problem. See `mlbench.friedman1` for details and references.

Usage

```
gen_friedman1(n = 100, nx = 10, sigma = 0.1)
```

Arguments

`n`	Integer specifying the number of samples (i.e., rows) to generate. Default is 100.
`nx`	Integer specifying the number of predictor variables to generate. Default is 10. Note that `nx` >= 5.
`sigma`	Numeric specifying the standard deviation of the standard Gaussian noise.

Value

A data frame with `n` rows and `nx` + 1 columns (for `nx` features and the response).

Examples

```
set.seed(2319)  # for reproducibility
friedman1 <- gen_friedman1(nx = 5)
pairs(friedman1, col = "purple2")
```

`gen_mease` *Generate data from the Mease model*

Description

Generate binary classification data from the Mease model Mease et al. (2007).

Usage

```
gen_mease(n = 1000, nsim = 1)
```

Arguments

`n`	Integer specifying the number of observations. Default is `1000`.
`nsim`	Integer specifying the number of binary repsonses to generate. Default is `1`.

Value

A data frame with `3 + nsim` columns. The first two columns give the values of the numeric features `x1` and `x2`. The third column (`yprob`) gives the true probabilities (i.e., PrY = 1 | X = x). The remaining `nsim` columns (`yclass<i>`, `i = 1, 2, ..., nsim`) give the simulated binary outcomes corresponding to `yprob`.

References

Mease D, Wyner AJ, Buja A. Boosted classification trees and class probability quantile estimation. Journal of Machine Learning Research. 2007; 8:409–439.

Examples

```
# Generate N = 1000 observations from the Mease model
set.seed(2254)  # for reproducibility
mease <- gen_mease(1000, nsim = 1)

# Plot predictor values colored by binary outcome
cols <- palette.colors(2, palette = "Okabe-Ito", alpha = 0.3)
plot(x2 ~ x1, data = mease, col = cols[mease$yclass1 + 1], pch = 19)
```

guide_setup *Generate GUIDE input files*

Description

Just a simple helper function I found useful while using the GUIDE terminal application (http://pages.stat.wisc.edu/~loh/guide
It creates two input text files required by GUIDE: a data file and description file.

Usage

```
guide_setup(
  data,
  path,
  dv = NULL,
  var.roles = NULL,
  na = "NA",
  file.name = NULL,
  data.loc = NULL,
  verbose = FALSE
)
```

Arguments

`data`	A data frame containing the training data.
`path`	Character string specifying the full path to where the GUIDE input files will be written to. If the given path does not exist, it will be created automatically using `dir.create()`.
`dv`	Character string specifying which column represents the target/ dependent variable.
`var.roles`	A named character vector specifying the role of each column.
`na`	Character string specifying the missing value indicator.
`file.name`	Character string giving the file name (or prefix) to use for the generated input files. If `NULL`, the default, it will be parsed from the `data` argument.
`data.loc`	Character string specifying the the full path to the data input file, which is used for the first line of the generated description file. If `NULL`, the default, it will be determined automatically by `path` and `file.name`. This is useful if the data input file does not reside in the same directory as the GUIDE executable.
`verbose`	Logical indicating whether or not to print progress information.

Value

No return value, only called for side effects; in this case, two text file are created for consumption by the GUIDE terminal application

Note

This function assumes that the GUIDE executable is located in the same directory specified by the ‘path‘ argument. For details, see the official software manual for GUIDE: http://pages.stat.wisc.edu/~loh/treeprogs/guide/guideman.pdf. This function has only been tested on GUIDE v38.0.

Examples

```
## Not run:
# New York air quality measurements
aq <- airquality
aq <- aq[!is.na(aq$Ozone), ]  # remove rows with missing response values

# Default variable roles
guide_setup(aq, path = "some/path/aq", dv = "Ozone")

# User specified variable roles
var.roles <- c("Ozone" = "d", "Solar.R" = "n", "Wind" = "n", "Temp" = "c",
               "Month" = "p", "Day" = "p")
guide_setup(aq, path = "some/path/aq", var.roles = var.roles)

## End(Not run)
```

`hitters` *Baseball data (corrected)*

Description

Major League Baseball data from the 1986 and 1987 seasons.

Format

A data frame with 322 observations of major league players on the following 20 variables.

AtBat Number of times at bat in 1986.

Hits Number of hits in 1986.

HmRun Number of home runs in 1986.

Runs Number of runs in 1986.

RBI Number of runs batted in in 1986.

Walks Number of walks in 1986.

Years Number of years in the major leagues.

CAtBat Number of times at bat during his career.

CHits Number of hits during his career.

CHmRun Number of home runs during his career.

CRuns Number of runs during his career.

CRBI Number of runs batted in during his career.

CWalks Number of walks during his career.

League A factor with levels `A` and `N` indicating player's league at the end of 1986.

Division A factor with levels `E` and `W` indicating player's division at the end of 1986.

PutOuts Number of put outs in 1986.

Assists Number of assists in 1986.

Errors Number of errors in 1986.

Salary 1987 annual salary on opening day in thousands of dollars.

NewLeague A factor with levels `A` and `N` indicating player's league at the beginning of 1987.

Note

This is a corrected version of the `Hitters` data set based on the corrections listed in Hoaglin and Velleman (1995).

References

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) *An Introduction to Statistical Learning with applications in R*, `https://www.statlearning.com`, Springer-Verlag, New York.

Hoaglin, D. C. and Velleman, P. F. (1995). *A critical look at some analyses ofmajor league baseball salaries*. The American Statistician, 49(3):277-285.

See Also

`Hitters`.

Examples

```
summary(hitters <-  treemisc::hitters)
plot(log(Salary) ~ Years, data = hitters)
```

`isle_post` *Importance sampled learning ensemble*

Description

Uses `glmnet` or `cv.glmnet` to fit the entire LASSO path for post-processing the individual trees of a tree-based ensemble (e.g., a random forest).

Usage

```
isle_post(
  X,
  y,
  newX = NULL,
  newy = NULL,
  cv = FALSE,
  nfolds = 5,
  family = NULL,
  loss = "default",
  offset = NULL,
  ...
)
```

Arguments

`X`	A matrix of training predictions, one column for each tree in the ensemble.
`y`	Vector of training response values. See `glmnet` for acceptable values (e.g., numeric for `family = "gaussian"`).
`newX`	Same as argument `X`, but should correspond to an independent test set. (Required whenever `cv = FALSE`.)
`newy`	Same as argument `y`, but should correspond to an independent test set. (Required whenever `cv = FALSE`.)
`cv`	Logical indicating whether or not to use n-fold cross-validation. Default is `FALSE` (Must be `TRUE` whenever `newX = NULL` and `newy = NULL`.)
`nfolds`	Integer specifying the number of folds to use for cross-validation (i.e., whenever `cv = TRUE`). Default is `FALSE`.
`family`	The model fitting family (e.g., `family = "binomial"` for binary outcomes); see `glmnet` for details on acceptable values.
`loss`	Optional character string specifying the loss to use for n-fold cross-validation. Default is `"default"`; see `cv.glmnet` for details. (Only used when `cv = TRUE`.)
`offset`	Optional value for the offset. Default is `NULL`, which corresponds to no offset.
`...`	Additional (optional) arguments to be passed on to `glmnet` (e.g., `intercept = FALSE`).

Value

A list with two components:

`results` A data frame with one row for each value of lambda in the coefficient path and columns giving the corresponding number of trees/non-zero coefficients, error metric(s), and the corresponding value of lambda.

lasso.fit The fitted `glmnet` or `cv.glmnet` object.

`ladboost` *Gradient tree boosting with least absolute deviation (LAD) loss*

Description

A poor-man's implementation of stochastic gradient tree boosting with LAD loss.

Print basic information about a fitted `"ladboost"` object.

Compute predictions from an `"ladboost"` object using new data.

Usage

```
ladboost(
  X,
  y,
  ntree = 100,
  shrinkage = 0.1,
  depth = 6,
  subsample = 0.5,
  init = median(y)
)

## S3 method for class 'ladboost'
print(x, ...)

## S3 method for class 'ladboost'
predict(object, newdata, ntree = NULL, individual = FALSE, ...)
```

Arguments

`X`	A data frame of only predictors.
`y`	A vector of response values
`ntree`	Integer specifying the number of trees in the ensemble to use. Defaults to using all the trees in the ensemble.
`shrinkage`	Numeric specifying the shrinkage factor.
`depth`	Integer specifying the depth of each tree.
`subsample`	Numeric specifying the proportion of the training data to randomly sample before building each tree. Default is `0.5`.
`init`	Numeric specifying the initial value to boost from. Defaults to the median response (i.e., `median(y)`).
`x`	An object of class `"ladboost"`.
`...`	Additional optional arguments. (Currently ignored.)
`object`	An object of class `"ladboost"`.
`newdata`	Data frame of new observations for making predictions.

`individual`	Logical indicating whether or not to return the (shrunken) predictions from each tree individually (`TRUE`) or the overall ensemble prediction (`FALSE`). Default is `FALSE`.

Value

An object of class `"ladboost"` which is just a list with the following components:

- `trees` A list of length `ntree` containing the individual `rpart` tree fits.
- `shrinkage` The corresponding shrinkage parameter.
- `depth` The maximum depth of each tree.
- `subsample` The (row) subsampling rate.
- `init` The initial constant fit.

A vector (`individual = TRUE`) or matrix (`individual = FALSE`) of predictions.

Note

By design, the final model does not include the predictions from the initial (constant) fit. So the constant is stored in the `init` component of the returned output to be used later by `predict.ladboost()`.

Examples

```
# Simulate data from the Friedman 1 benchmark problem
set.seed(1025)  # for reproducibility
trn <- gen_friedman1(500)  # training data
tst <- gen_friedman1(500)  # test data

# Gradient boosted decision trees
set.seed(1027)  # for reproducibility
bst <- ladboost(subset(trn, select = -y), y = trn$y, depth = 2)
pred <- predict(bst, newdata = tst)
mean((pred - tst$y) ^ 2)
```

`lift` *Gain and lift charts*

Description

Validates predicted probabilities against a set of observed (binary) outcomes.

Usage

```
lift(prob, y, pos.class = NULL, cumulative = TRUE, nbins = 0)

## S3 method for class 'lift'
plot(
  x,
  refline = TRUE,
  refline.col = 2,
  refline.fill = 2,
  refline.lty = "dashed",
  refline.lwd = 1,
  ...
)
```

Arguments

`prob`	Vector of predicted probabilities.
`y`	Vector of binary (i.e., 0/1) outcomes. If `y` is coded as anything other than 0/1, then you must specify which of the two categories represents the "positive" class (i.e., the class for which the probabilities specified in `prob` correspond to) via the `pos.class` argument.
`pos.class`	Numeric/character string specifying which values in `y` correspond to the "positive" class. Default is `NULL`. (Must be specified whenever `y` is not coded as 0/1, where 1 is assumed to represent the "positive" class.)
`cumulative`	Logical indicating whether or not to compute cumulative lift (i.e., gain). Default is `TRUE`.
`nbins`	Integer specifying the number of bins to use when computing lift. Default is 0, which corresponds to no binning. For example, setting `nbins = 10` will result in computing lift within each decile of the sorted probabilities.
`x`	An object of class `"lift"`.
`refline`	Logical indicating whether or not to include a reference line.
`refline.col`	The color to use for the reference line. Default is `2`.
`refline.fill`	The color to use for filling in the polygon-shaped reference line. Default is `2`.
`refline.lty`	The type of line to use for the reference line. Default is `"dashed"`.
`refline.lwd`	The width of the reference line. Default is `1`.
`...`	Additional optional argument to be passed on to other methods.

Value

A `"lift"` object, which is essentially a list with the following components:

`"lift"` A numeric vector containing the computed lift values.

`"prop"` The corresponding proportion of cases associated with each lift value.

`"cumulative"` Same value as that supplied via the `cumulative` argument. (Used by the `plot.lift()` method.)

`load_eslmix` *Gaussian mixture data*

Description

Load the Gaussian mixture data from Hastie et al. (2009, Sec. 2.3.3).

Usage

```
load_eslmix()
```

Value

A list with the following components:

x 200 x 2 matrix of training predictors.

y class variable; logical vector of 0s and 1s - 100 of each.

xnew matrix 6831 x 2 of lattice points in predictor space.

prob vector of 6831 probabilities (of class TRUE) at each lattice point.

marginal marginal probability at each lattice point.

px1 69 lattice coordinates for `x1`.

px2 99 lattice values for `x2` (69*99=6831).

means 20 x 2 matrix of the mixture centers, first ten for one class, next ten for the other.

Source

https://web.stanford.edu/~hastie/ElemStatLearn/datasets/mixture.example.info.txt

References

Trevor Hastie, Robert. Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics. Springer-Verlag, 2009.

Examples

```
eslmix <- load_eslmix()
names(eslmix)
```

lsboost — *Gradient tree boosting with least squares (LS) loss*

Description

A poor-man's implementation of stochastic gradient tree boosting with LS loss.

Print basic information about a fitted "lsboost" object.

Compute predictions from an "lsboost" object using new data.

Usage

```
lsboost(
  X,
  y,
  ntree = 100,
  shrinkage = 0.1,
  depth = 6,
  subsample = 0.5,
  init = mean(y)
)

## S3 method for class 'lsboost'
print(x, ...)

## S3 method for class 'lsboost'
predict(object, newdata, ntree = NULL, individual = FALSE, ...)
```

Arguments

`X`	A data frame of only predictors.
`y`	A vector of response values
`ntree`	Integer specifying the number of trees in the ensemble to use. Defaults to using all the trees in the ensemble.
`shrinkage`	Numeric specifying the shrinkage factor.
`depth`	Integer specifying the depth of each tree.
`subsample`	Numeric specifying the proportion of the training data to randomly sample before building each tree. Default is `0.5`.
`init`	Numeric specifying the initial value to boost from. Defaults to the mean response (i.e., `mean(y)`).
`x`	An object of class "lsboost".
`...`	Additional optional arguments. (Currently ignored.)
`object`	An object of class "lsboost".
`newdata`	Data frame of new observations for making predictions.

`individual` Logical indicating whether or not to return the (shrunken) predictions from each tree individually (`TRUE`) or the overall ensemble prediction (`FALSE`). Default is `FALSE`.

Value

An object of class `"lsboost"` which is just a list with the following components:

- `trees` A list of length `ntree` containing the individual `rpart` tree fits.
- `shrinkage` The corresponding shrinkage parameter.
- `depth` The maximum depth of each tree.
- `subsample` The (row) subsampling rate.
- `init` The initial constant fit.

A vector (`individual = TRUE`) or matrix (`individual = FALSE`) of predictions.

Note

By design, the final model does not include the predictions from the initial (constant) fit. So the constant is stored in the `init` component of the returned output to be used later by `predict.lsboost()`.

Examples

```
# Simulate data from the Friedman 1 benchmark problem
set.seed(1025)  # for reproducibility
trn <- gen_friedman1(500)  # training data
tst <- gen_friedman1(500)  # test data

# Gradient boosted decision trees
set.seed(1027)  # for reproducibility
bst <- lsboost(subset(trn, select = -y), y = trn$y, depth = 2)
pred <- predict(bst, newdata = tst)
mean((pred - tst$y) ^ 2)
```

`mushroom` *Mushroom edibility*

Description

Mushrooms described in terms of physical characteristics.

Format

A data frame with 8124 rows and 23 variables.

Source

Knopf, Alfred A. *The Audubon Society Field Guide to North American Mushrooms*, G. H. Lincoff (Pres.), 1981.

predict.rforest *Random forest predictions*

Description

Compute predictions from an "rftree" object.

Usage

```
## S3 method for class 'rforest'
predict(object, newX, predict.all = FALSE, ...)
```

Arguments

object	An object of class "rftree".
newX	Data frame or matrix of new feature values.
predict.all	Logical indicating whether or not to return predictions for each individual tree. Default is FALSE, which only returns the aggregated predictions.
...	Additional optional arguments. (Currently ignored.)

Value

A vector of predictions. For binary outcomes coded as 0/1, the predictions represent Pr(Y = 1).

proximity *Proximity matrix*

Description

Compute proximity matrix from a random forest or matrix of terminal node assignments (one row for each observation and one column for each tree in the forest).

Usage

```
proximity(x, ...)

## Default S3 method:
proximity(x, sparse = NULL, upper = TRUE, ...)

## S3 method for class 'matrix'
proximity(x, sparse = NULL, upper = TRUE, ...)

## S3 method for class 'ranger'
proximity(x, data = NULL, sparse = NULL, upper = TRUE, ...)
```

Arguments

`x`	Either a `ranger` object or a matrix of terminal node assignments (one row for each observation and one column for each tree in the forest).
`...`	Additional optional argument. (Currently ignored.)
`sparse`	Logical or `NULL` indicating whether or not the resulting matrix should be sparse. If `NULL` (the default) it is made sparse when more than half the entries are 0.
`upper`	Logical indicating whether or not to return the proximities in upper triangular form (`TRUE`) or as a symmetric matrix (`FALSE`). Default is `TRUE`.
`data`	Optional data frame passed on to `predict.ranger`. It's a good idea to pass the data via this argument whenever `x` is a `ranger` object. If `NULL` (the default) it will be looked for recursively.

Value

A matrix or sparse Matrix (`sparse = TRUE`) of pairwise proximity (i.e., similarity) scores between training observations.

`prune_se` *Prune an* `rpart` *object*

Description

Prune an `rpart` object using the standard error (SE) of the cross-validation results.

Usage

```
prune_se(object, prune = TRUE, se = 1)
```

Arguments

`object`	An object that inherits from class `"rpart"`.
`prune`	Logical indicating whether or not to return the pruned decision tree. Default is `TRUE`. If `FALSE`, the optimal value of the cost-complexity parameter is returned instead.
`se`	Numeric specifying the number of standard errors to use when pruning the tree. Default is `1`, which corresponds to the 1-SE rule described in Breiman et al. (1984).

Value

Either an object that inherits from class `"rpart"` (ideally, one that's been simplified using cost-complexity pruning with the 1-SE rule) or a numeric value representing the cost-complexity parameter to use for pruning.

References

Breiman, L., Friedman, J., and Charles J. Stone, R. A. O. (1984). Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

See Also

`prune`

`rforest` *Random forest*

Description

A poor man's implementation of random forest (Breiman, 2001) with the option to incorporate random rotations as described in Blaser and Fryzlewicz (2016).

Usage

```
rforest(X, y, mtry = NULL, ntree = 500, rotate = FALSE, ...)
```

Arguments

`X`	A data frame or a matrix of predictors.
`y`	Numeric vector of response value. For binary outcomes, `y` should be mapped to {0, 1}. Note that multiclass outcomes are not supported.
`mtry`	Integer specifying the number of variables randomly sampled as candidates splitters at each node in a tree. Note that the default values are different for classification (`floor(sqrt(p))` where `p` is number of columns of `X`) and regression `floor(p/3)`).
`ntree`	Integer specifying the number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times. Default is `500`.
`rotate`	Logical indicating whether or not to randomly rotate the feature values prior to fitting each tree. Default is `FALSE` which results in a traditional random forest.
`...`	Optional arguments to be passed on to `randomForest` (e.g., `nodesize = 10`).

Value

An object of class `"rforest"`, which is essentially a list of rftree objects.

References

Breiman, Leo. (2001), Random Forests, Machine Learning 45(1), 5-32.

Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. Journal of Machine Learning Research, 17:1–26, 2016.

`rrm` *Random rotation matrix*

Description

Generates a random rotation matrix as described in Blaser and Fryzlewicz (2016).

Usage

```
rrm(n)
```

Arguments

`n`	Integer specifying the dimension of the resulting (square) random rotation matrix.

Value

An `n`-by-`n` matrix (i.e., an object of class `c("matrix" "array")`.

Source

https://www.jmlr.org/papers/volume17/blaser16a/blaser16a.pdf.

References

Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. Journal of Machine Learning Research, 17:1–26, 2016.

Examples

```
(R <- rrm(3))
det(R)  # determinant should always be +1
solve(R)  # R^{-1} = R'
t(R)  # R^{-1} = R'
```

`tree_diagram` *Tree diagram*

Description

Draw a decision tree diagram from a fitted `rpart` model.

Usage

```
tree_diagram(object, prob = TRUE, box.palette = "BuOr", ...)
```

Arguments

`object`	An `rpart` object.
`prob`	Logical indicating whether or not to display leaf node probability estimates for classification trees; default is `TRUE`.
`box.palette`	Chjaracter string specifying the palette to use for coloring the nodes; see `rpart.plot` for details.
`...`	Additional optional argumebts to be passed onto `rpart.plot`.

Value

No return value, only called for side effects; in this case, a decision tree diagram constructed by a simple call to `rpart.plot`.

Note

This function is just a light wrapper around `rpart.plot` and was used to produce several of the tree diagrams in the accompanying book.

`wilson_hilferty` *Modified Wilson-Hilferty approximation*

Description

Implements the modified Wilson-Hilferty (1931) approximation used by GUIDE to convert a chi-square random variable to an approximate chi-square random variable with one degree of freedom.

Usage

```
wilson_hilferty(x, df)
```

Arguments

`x`	Numeric value of the observed chi-square statistic.
`df`	Integer specifying the degrees of freedom associated with `x`.

Value

An approximate chi-square statistic with one degree of freedom.

Examples

```
wilson_hilferty(2056, df = 4)
wilson_hilferty(1831, df = 20)

set.seed(1144)  # for reproducibility
x <- rchisq(1000, df = 10)
w <- sapply(x, FUN = function(x) wilson_hilferty(x, df = 10))
px <- pchisq(x, df = 10)
pw <- pchisq(round(w), df = 1)
plot(px, pw)
abline(0, 1, lty = 2, col = 2)
cor(px, pw)
```

`wine` *Wine quality*

Description

These data are related to red and white variants of the Portuguese "Vinho Verde" wine. For details, see Cortez et al. (2009). Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g., there is no data about grape types, wine brand, wine selling price, etc.). These data can be used for classification or regression tasks. The classes are ordered and not balanced (e.g., there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, it is not known if all input variables are relevant. So it could be interesting to test feature selection methods.

Format

A data frame with 6497 rows and 13 variables.

Details

fixed.acidity Input feature (continuous).

volatile.acidity Input feature (continuous).

citric.acid Input feature (continuous).

residual.sugar Input feature (continuous).

chlorides Input feature (continuous).

free.sulfur.dioxide Input feature (continuous).

total.sulfur.dioxide Input feature (continuous).

density Input feature (continuous).

pH Input feature (continuous).

sulphates Input feature (continuous).

alcohol Input feature (continuous).

quality Target variable (continuous).

type Input feature specifying whether it's a red or white wine (factor).

Source

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

`xy_grid` *Create a Cartesian product from evenly spaced values of two variables*

Description

Create a Cartesian product from evenly spaced values of two variables.

Usage

```
xy_grid(x, ...)

## Default S3 method:
xy_grid(x, y, grid.resolution = 51, col.names = NULL, ...)

## S3 method for class 'formula'
xy_grid(x, data, grid.resolution = 51, ...)

## S3 method for class 'matrix'
xy_grid(x, grid.resolution = 51, ...)

## S3 method for class 'data.frame'
xy_grid(x, grid.resolution = 51, ...)
```

Arguments

`x`	Either a numeric vector (if argument `y` is also specified), a matrix-like object (e.g., a data frame), or two-variable formula of the form `y ~ x`.
`...`	Additional (optional) arguments. (Currently ignored.)
`y`	A numeric vector representing the second variable (only required if `x` is a numeric vector).
`grid.resolution`	Integer specifying the number of equally-spaced values to use for each numeric variable. For example, if `grid.resolution = k`, then the final data frame will have `k^2` rows (formed by a Cartesian product).
`col.names`	Optional vector of column names to use for the output whenever both `x` and `y` are supplied.
`data`	A data frame containing the variables specified in `x` if `x` is a formula.
`formula`	A two-variable formula of the form `y ~ x`. The response (i.e., the variable on the left side of the formula) corresponds to the second column of the output.

Value

A data frame representing the Cartesian product between equally spaced values from each variable.

Examples

```
x1 <- 1:3
x2 <- letters[1L:3L]
xy_grid(x1, x2, gr = 3, col.names = c("x1", "x2"))  # will have 3^2=9 rows
xy_grid(m <- cbind(x1, x2), gr = 3)     # equivalent
xy_grid(d <- as.data.frame(m), gr = 3)  # equivalent
xy_grid(x2 ~ x1, data = d, gr = 3)      # equivalent
```

Index