Package 'opencesp'

June 1, 2026

Type Package

Title Generation and Evaluation of Synthetic Tabular Datasets

Version 0.4.0

Maintainer Rémy Chapelle <remy.chapelle@universite-paris-saclay.fr>

Description Various tools developed as part of the Open-CESP (Centre de recherche en Epidémiologie et Santé des Populations) initiative to generate and evaluate synthetic datasets for statistical disclosure control.
This includes tools to investigate the risk-utility tradeoff achievable with given synthesis methods, as well as statistical tools to estimate (conditional) probability distributions. The main eventual aim is to help researchers and statisticians disseminate open research data.

License GPL-3

URL https://opencesp.vjf.inserm.fr/en

Encoding UTF-8

NeedsCompilation yes

Imports stats, cluster, rpart, parallel, fastmap, PCAmixdata, randomForest, mice

Suggests gbm

RoxygenNote 8.0.0

Language en-US

Author Rémy Chapelle [aut, cre] (ORCID: <https://orcid.org/0009-0006-3088-0354>), Centre de recherche en Epidémiologie et Santé des Populations [cph]

Repository CRAN

Date/Publication 2026-06-01 08:30:09 UTC

Contents

`opencesp-package` *opencesp: Generation and Evaluation of Synthetic Tabular Datasets*

Description

Various tools developed as part of the Open-CESP (Centre de recherche en Epidémiologie et Santé des Populations) initiative to generate and evaluate synthetic datasets for statistical disclosure control. This includes tools to investigate the risk-utility tradeoff achievable with given synthesis methods, as well as statistical tools to estimate (conditional) probability distributions. The main eventual aim is to help researchers and statisticians disseminate open research data.

Author(s)

Maintainer: Rémy Chapelle <remy.chapelle@universite-paris-saclay.fr> (ORCID)

Authors:

- Rémy Chapelle <remy.chapelle@universite-paris-saclay.fr> (ORCID)

Other contributors:

- Centre de recherche en Epidémiologie et Santé des Populations [copyright holder]

See Also

Useful links:

- https://opencesp.vjf.inserm.fr/en

`adaptive_matches_prop` *Adaptive proportion of matches*

Description

`adaptive_matches_prop` returns the proportion of rows in an original dataframe that can be found in a synthetic dataframe.

Usage

```
adaptive_matches_prop(orig, synth)
```

Arguments

`orig`	A dataframe containing original data.
`synth`	A dataframe containing synthetic data, with the same variables as `orig`.

Details

Matching is based on Gower's distance and is adaptive: an original row is counted as matched when its nearest synthetic row is closer than its nearest other original row.

Value

The proportion.

Examples

```
data(iris)

orig <- iris

synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

adaptive_matches_prop(orig, synth)
```

ASDED *Average squared differences between empirical distributions*

Description

ASDED returns an empirical distributions-based metric, as proposed in doi:10.29012/jpc.v1i1.568 (see equation 5 in this paper).

Usage

```
ASDED(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The ASDED metric.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

ASDED(orig, synth)
```

`avatarize` *Avatarization of a dataset*

Description

`avatarize` returns a dataframe containing avatars of original observations, as described in doi:10.1038/s41746023007715.

Usage

```
avatarize(data, k, npc)
```

Arguments

`data`	A dataframe containing original data.
`k`	The number of nearest neighbors to consider.
`npc`	The number of dimensions to use for dimensionality reduction.

Details

Avatarization being a stochastic procedure, this function does not necessarily give the same result for successive calls with the same parameters. The distance metric used in this implementation of the Avatar method is the Euclidean distance.

Value

The avatar dataframe.

Examples

```
data(iris)

orig <- iris[sapply(iris, is.numeric)]

synth <- avatarize(orig, k = 5, npc = 2)

plot(orig$Sepal.Width, orig$Sepal.Length)
points(synth$Sepal.Width, synth$Sepal.Length, col = "red")
```

CCM_RS *Cross-classification metric in the real-synthetic order*

Description

CCM_RS returns the CrCl-RS metric, as described in doi:10.1186/s12874020009771. This metric is calculated as the average, over categorical variables, of the ratio of the classification accuracy on the original hold-out data to the classification accuracy on the synthetic data. The classifiers used are CARTs. The training and hold-out sets respectively correspond to the first and second half of the original data.

Usage

```
CCM_RS(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The CrCl-RS metric.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

CCM_RS(orig, synth)
```

CCM_SR *Cross-classification metric in the synthetic-real order*

Description

CCM_SR returns the CrCl-SR metric, as described in doi:10.1186/s12874020009771.

Usage

```
CCM_SR(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The CrCl-SR metric.

See Also

CCM_RS()

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

CCM_SR(orig, synth)
```

---

cor_F1 *Membership F1 metric*

Description

cor_F1 returns a corrected F1 membership disclosure metric, as described in doi:10.1093/jamiaopen/ooac083.

Usage

```
cor_F1(orig_train, orig_ho, synth, m, h, N, t_prop = nrow(orig_train)/N)
```

Arguments

orig_train	A subset of the original dataframe to use for training.
orig_ho	The hold-out subset of the original dataframe, with the same variables as orig_train.
synth	A dataframe containing data synthesized from orig_train, with the same variables as orig_train and orig_ho.
m	The desired number of records in the attack dataset.
h	The Hamming distance threshold to use.
N	The size of the source population from which original data originate.
t_prop	The desired proportion of records in the attack dataset originating from orig_train.

Value

The membership disclosure metric, or NA if the F1 score is undefined.

Examples

```
data(iris)

orig <- iris
ids_train <- sample(1:nrow(orig), round(nrow(orig) * 0.8))
orig_train <- orig[ids_train, ]
orig_ho <- orig[-ids_train, ]

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig_train, function(x) {
                          x[sample(length(x), nrow(orig_train), replace = TRUE)]
                       }))

cor_F1(orig_train, orig_ho, synth, m = 20, h = 2, N = 1000)
```

CRM_RS *Cross-regression metric in the real-synthetic order*

Description

CRM_RS returns a cross-regression metric, adapted from the CrCl-RS metric described in doi:10.1186/s12874020009771. This metric is calculated as the average, over numeric variables, of the ratio between the mean squared prediction error obtained on the original hold-out data and the mean squared prediction error obtained on the synthetic data. Regressions are performed using CART. The training and hold-out sets respectively correspond to the first and second half of the original data.

Usage

```
CRM_RS(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The cross-regression metric.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

CRM_RS(orig, synth)
```

CRM_SR *Cross-regression metric in the synthetic-real order*

Description

CRM_SR returns a cross-regression metric, adapted from the CrCl-SR metric described in doi:10.1186/s12874020009771.

Usage

```
CRM_SR(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The cross-regression metric.

See Also

CRM_RS()

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

CRM_SR(orig, synth)
```

dcr *Distance to the closest record*

Description

dcr returns the median distance between each synthetic unit and its nearest original neighbor. This type of metric was notably suggested in doi:10.14778/3231751.3231757. Here, Gower's distance is used to account for mixed data types; see, for example, doi:10.48550/arXiv.2101.02481.

Usage

```
dcr(orig, synth, method = c("auto", "euclidean", "gower"))
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.
method	One of "auto", "euclidean" or "gower".

Value

The median distance.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

dcr(orig, synth)
```

dep_order *Dependency order in a data set*

Description

dep_order returns the dependency order of variables determined by hierarchical clustering.

Usage

```
dep_order(
  df,
  blocks = rep(1, ncol(df)),
  score_func,
  bf_first = TRUE,
  n_bf = 1,
  n_cores = 1
)
```

Arguments

df	A dataframe.
blocks	A numeric vector containing indices of independent variable blocks.
score_func	A custom scoring function, taking exactly two parameters: x (a dataframe, or NULL) and y (a vector of the same size as x), and returning a numeric value. If x is null, the function must return a marginal score, else a conditional score (of y given x).
bf_first	A boolean. If TRUE, the algorithm will try each variable as the first variable in the order and keeps the best-scoring result (in each block).
n_bf	An integer, the number of clusters left unmerged by the hierarchical clustering procedure, and reordered by exhaustive search (in each block of blocks).
n_cores	An integer, the number of CPU cores to use.

Value

A numeric vector representing the order of dependency.

Examples

```
data(iris)

iris <- iris[sapply(iris, is.numeric)]

score_lin <- function(x, y) {
  if(is.null(x)) resids <- y-mean(y)
  else {
    df <- cbind.data.frame(y, x)
    fit <- lm(y ~ ., data = df)
    resids <- fit$residuals
  }
  sd_hat <- sd(resids)
  return(sum(dnorm(resids, sd = sd_hat, log = TRUE)))
}

dep_order(iris, score_func = score_lin)
```

GCAP *Generalized Correct Attribution Probability (GCAP)*

Description

GCAP returns the GCAP for a given row of a dataframe, as described in doi:10.48550/arXiv.2310.06571.

Usage

```
GCAP(row, data, keys, rad_keys, targets, rad_targets)
```

Arguments

row	A row of a dataframe.
data	The dataframe.
keys	A vector containing the names of the variables of the dataframe to take as keys.
rad_keys	A vector containing radii, one for each numeric key. The order of the radii must match the order of the variables in the dataframe.
targets	A vector containing the names of the variables of the dataframe to take as targets.
rad_targets	A vector containing radii, one for each numeric target. The order of the radii must match the order of the variables in the dataframe.

Value

The computed probability.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

target_row <- orig[1, ]

keys <- colnames(orig)[1:3]
targets <- colnames(orig)[4:5]

radii_k <- sapply(Filter(is.numeric, orig[, keys]), function(x) {
  d <- diff(sort(x))
  mean(pmin(c(Inf, d), c(d, Inf)))
})

radii_t <- sapply(Filter(is.numeric, orig[, targets]), function(x) {
  d <- diff(sort(x))
```

```
  mean(pmin(c(Inf, d), c(d, Inf)))
})

GCAP(target_row, synth, keys, radii_k, targets, radii_t)
```

---

get_prec *Precision of numeric variables*

Description

`get_prec` returns the number of decimal places found in a numeric vector.

Usage

```
get_prec(vec, max = TRUE)
```

Arguments

`vec`	A numeric vector.
`max`	A boolean. If `TRUE`, only the maximum precision is returned.

Value

Either a single integer or an integer vector.

Examples

```
data(iris)

sapply(iris, get_prec)
```

GTCAP *Generalized Targeted Correct Attribution Probability (GTCAP)*

Description

`GTCAP` computes the GTCAP for a synthetic version of a dataframe, as described in doi:10.48550/arXiv.2310.06571.

Usage

```
GTCAP(orig, synth, keys, rad_keys, targets, rad_targets, n_cores = 1)
```

Arguments

orig	The original dataframe.
synth	The synthetic dataframe.
keys	A vector containing names of variables of the dataframe to take as keys.
rad_keys	A vector containing radii, one for each numeric key. The order of the radii must match the order of the variables in the dataframe.
targets	A vector containing names of variables of the dataframe to take as targets.
rad_targets	A vector containing radii, one for each numeric target. The order of the radii must match the order of the variables in the dataframe.
n_cores	The number of logical processes to use for the computation.

Value

A list with the following elements:

mean The standardized mean GCAP for target synthetic rows.

ind A vector containing standardized GCAP values, one for each target synthetic row.

Examples

```
data(iris)

orig <- iris[1:25, ] # we use only a sample of the data because GTCAP is computationally demanding

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

target_row <- orig[1, ]

keys <- colnames(orig)[1:3]
targets <- colnames(orig)[4:5]

radii_k <- sapply(Filter(is.numeric, orig[, keys]), function(x) {
  d <- diff(sort(x))
  mean(pmin(c(Inf, d), c(d, Inf)))
})

radii_t <- sapply(Filter(is.numeric, orig[, targets]), function(x) {
  d <- diff(sort(x))
  mean(pmin(c(Inf, d), c(d, Inf)))
})

GTCAP(orig, synth, keys, radii_k, targets, radii_t)$mean
```

hellinger_distance *Estimated Hellinger distance*

Description

hellinger_distance returns an estimate of the Hellinger distance between original and synthetic data, as suggested (for example) in doi:10.1109/ACCESS.2022.3144765. Missing values in the input vectors are ignored.

Usage

```
hellinger_distance(orig, synth)
```

Arguments

orig	A vector containing original data.
synth	A vector containing synthetic data, of the same type as orig.

Value

The estimated Hellinger distance.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

hellinger_distance(orig$Species, synth$Species)
```

impute_rf *Random-forest imputation*

Description

impute_rf imputes missing values in a dataframe using the random-forest method from mice.

Usage

```
impute_rf(df, ...)
```

Arguments

df	A dataframe.
...	Additional arguments passed to mice::mice().

Value

A dataframe with imputed values.

Examples

```
data(airquality)

colSums(is.na(airquality))

airquality <- impute_rf(airquality)

colSums(is.na(airquality))
```

---

ind_blocks *Independent blocks of variables*

Description

ind_blocks finds approximately independent blocks of variables in a dataset by hierarchical clustering.

Usage

```
ind_blocks(df, crit = c("dim", "n.int"), max_size = ncol(df)/2, n_cores = 1)
```

Arguments

df	A dataframe.
crit	One of "dim" or "n.int". This specifies the interpretation of the parameter max_size.
max_size	The maximum size of the blocks returned. If crit = "dim", the size is in terms of the number of variables in each block. If crit = "n.int", the size takes into account the number of modalities of categorical variables.
n_cores	An integer, the number of CPU cores to use.

Value

An integer vector representing membership of each variable to the found blocks.

Examples

```
data(iris)

iris <- iris[sapply(iris, is.numeric)]

blocks <- ind_blocks(iris, max_size = 2)

score_lin <- function(x, y) {
  if(is.null(x)) resids <- y-mean(y)
  else {
    df <- cbind.data.frame(y, x)
    fit <- lm(y ~ ., data = df)
    resids <- fit$residuals
  }
  sd_hat <- sd(resids)
  return(sum(dnorm(resids, sd = sd_hat, log = TRUE)))
}

dep_order(iris, score_func = score_lin, blocks = blocks)
```

---

interval_overlap *Overlap of confidence intervals*

Description

`interval_overlap` returns the overlap of confidence intervals constructed from original and synthetic data, as described (for example) in doi:10.1111/rssa.12358. Intervals are constructed from t distributions in case of numeric data, and with the Clopper–Pearson method for binomial intervals in case of categorical data.

Usage

```
interval_overlap(orig, synth, cat = NULL, conf.level = 0.95)
```

Arguments

`orig`	A vector containing original data.
`synth`	A vector containing synthetic data, of the same type as `orig`.
`cat`	The category to consider to construct binomial intervals in case of categorical data.
`conf.level`	The desired level of confidence for the confidence intervals.

Value

The computed overlap.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

interval_overlap(orig$Sepal.Width, synth$Sepal.Width)
```

LCM *Log-cluster metric*

Description

LCM returns a log-cluster metric, as described in doi:10.1186/s12874020009771. This metric is based on cluster analysis as proposed initially in doi:10.29012/jpc.v1i1.568. Here we perform cluster analysis based on Gower's distance to account for mixed data types (see for example doi:10.48550/arXiv.2101.02481.

Usage

```
LCM(orig, synth, n_clusters)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.
n_clusters	The number of clusters to use.

Value

The log-cluster metric.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

LCM(orig, synth, 3)
```

matches_prop *Proportion of matches*

Description

matches_prop returns the proportion of rows in an original dataframe that can be found in a synthetic dataframe.

Usage

```
matches_prop(orig, synth, method = c("exact", "gower"), thr = NULL)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.
method	One of "exact" or "gower".
thr	A numeric threshold used to decide whether two rows match when method = "gower". If NULL, the threshold is chosen automatically (see Details).

Details

Duplicates within each dataframe are treated as a single record when method = "exact". If thr = NULL, the threshold is chosen automatically as the average nearest-neighbour distance between rows of orig. It represents the typical spacing between observations in the original data.

Value

The proportion.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

matches_prop(orig, synth)
```

`mean_hellinger` *Mean Hellinger distance*

Description

`mean_hellinger` returns the mean estimated Hellinger distance across all variables of original and synthetic datasets. This is close to doi:10.1093/jamia/ocaa249, but using the mean instead of the median.

Usage

```
mean_hellinger(orig, synth)
```

Arguments

`orig`	A dataframe containing original data.
`synth`	A dataframe containing synthetic data, with the same variables as `orig`.

Value

The average estimated Hellinger distance.

See Also

`hellinger_distance()`

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

mean_hellinger(orig, synth)
```

`outlier_coverage` *Outlier coverage*

Description

`outlier_coverage` returns the outlier coverage, which is originally a privacy metric that compares two marginal distributions. Here its average value across all variables in the original data is returned .

Usage

```
outlier_coverage(orig, synth)
```

Arguments

`orig`	A dataframe containing original data.
`synth`	A dataframe containing synthetic data, with the same variables as `orig`.

Value

The value of the metric.

See Also

SDMetrics documentation: OutlierCoverage

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

outlier_coverage(orig, synth)
```

outlier_learning_factor *Outlier learning factor*

Description

outlier_learning_factor returns the outlier learning factor, a privacy metric which aims at quantifying the tendency of a synthesizer to produce observations that lie in low-density regions of the original data. The metric is defined as the average distance between outliers (as defined by the user) and their nearest synthetic neighbor.

Usage

```
outlier_learning_factor(orig, outlier_ids, synth)
```

Arguments

orig	A dataframe containing original data.
outlier_ids	A numeric vector containing the indices of the outliers in orig.
synth	A dataframe containing synthetic data, with the same variables as orig.

Details

The distance used is Gower's distance.

Value

The value of the metric.

Examples

```
data(iris)

orig <- iris[sapply(iris, is.numeric)]

# find the 5% most atypical observations by a nearest-neighbor rule
D <- as.matrix(dist(orig, method = "euclidean"))
score <- apply(D, 1, function(x) sort(x)[2])
idx_outliers <- order(score, decreasing = TRUE)[seq_len(ceiling(0.05 * nrow(orig)))]

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

outlier_learning_factor(orig, idx_outliers, synth)
```

PCD_cat *Pairwise correlation difference adapted for categorical data*

Description

PCD_cat returns a metric adapted from the pairwise correlation difference suggested in doi:10.1186/s12874020009771. Contrary to the original proposal, the matrices from which the Frobenius norm is returned contain Cramer's V rather than Pearson correlation coefficients. Consequently, the metric is suitable to nominal data, and is computed on subsets of the input dataframes represented by all their categorical variables.

Usage

```
PCD_cat(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The adapted pairwise correlation difference.

Examples

```
data(iris)

# categorize numeric variables for the example
iris[] <- lapply(iris, function(x) if(is.numeric(x)) cut(x, 2) else x)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

PCD_cat(orig, synth)
```

PCD_num *Pairwise correlation difference*

Description

PCD_num returns the pairwise correlation difference, as defined in doi:10.1186/s12874020009771. The metric is computed on subsets of the input dataframes represented by all their numeric variables.

Usage

```
PCD_num(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

The pairwise correlation difference.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling with replacement
synth <- iris[sample(1:nrow(orig), nrow(orig), replace = TRUE), ]

PCD_num(orig, synth)
```

pgb *Parallel gradient boosting*

Description

pgb fits a PGB model.

Usage

```
pgb(formula, data, pvalid = 0.2, valid_data = NULL, M = 50, ...)
```

Arguments

formula	A formula, with a response but no interaction terms.
data	A dataframe, the training data.
pvalid	A number between 0 and 1, the proportion of observations that are sampled to form a validation set for early stopping.
valid_data	A dataframe (with the same structure as the training data) used as a validation set for early stopping.
M	An integer, the number of target quantiles.
...	Additional arguments passed to pgb_control().

Details

Note that if valid_data is provided, then pvalid must be 0.

Value

An object of class "pgb".

Examples

```
data(iris)

train_data <- iris[1:100, ]
valid_data <- iris[101:nrow(iris), ]

# train with a specified validation set
fit <- pgb(Sepal.Width ~ ., data = train_data, valid_data = valid_data, pvalid = 0)

# randomly select 20% of observations to form a validation set
fit <- pgb(Sepal.Width ~ ., data = iris, pvalid = 0.2)

# train without early stopping
fit <- pgb(Sepal.Width ~ ., data = iris, pvalid = 0, ntrees = 50)
```

---

pgb_control *Controls for PGB fits*

Description

pgb_control encapsulates the hyperparameters to be used for fitting PGB models.

Usage

```
pgb_control(
  ntrees = 10000,
  early_stopping_rounds = 100,
  maxdepth = 4,
  minbucket = 5,
  eta = 0.02,
  subsample = 0.5,
  maxbin = 256
)
```

Arguments

ntrees	An integer, the maximum number of trees to use.
early_stopping_rounds	An integer, the maximum number of iterations allowed without improvement of the validation error.
maxdepth	An integer, the maximum depth of the trees.
minbucket	An integer, the minimum number of training observations in each leaf of the trees.
eta	A positive number, the learning rate of the procedure.
subsample	A number between 0 and 1, the proportion of training observations to sample at each iteration.
maxbin	An integer, the maximum number of discrete bins to bucket continuous features. In the current implementation, this is also the number of observations sampled at each iteration to perform the line search.

Value

A list with the options.

Examples

```
data(iris)

controls <- pgb_control(ntrees = 1) # to fit a one-tree PGB model

# the following two lines are equivalent
fit <- do.call(pgb_cvh, c(list(formula = Sepal.Width ~ ., data = iris), controls))
fit <- pgb_cvh(Sepal.Width ~ ., data = iris, ntrees = 1)
```

pgb_cvh *Parallel gradient boosting with cross-validation*

Description

pgb_cvh fits a PGB model, with its hyperparameters determined by cross-validation

Usage

```
pgb_cvh(formula, data, nfolds = 5, select_h = c("greedy", "cv", "none"), ...)
```

Arguments

formula	A formula, with a response but no interaction terms.
data	A dataframe, the training data.
nfolds	An integer, the number of folds for the cross-validation procedure.
select_h	One of "none", "cv", or "greedy". If "cv", the smoothing bandwidth is also determined, by cross-validation. If "greedy", an heuristic is used where the number of iterations is determined by cross-validation, and the optimal smoothing bandwidth is determined for this number of iterations. If "none", the bandwidth is not determined.
...	Additional arguments to pgb().

Value

An object of class "pgb".

Examples

```
data(iris)

fit <- pgb_cvh(Sepal.Width ~ ., data = iris)
```

pMSE *Propensity score mean-squared error*

Description

pMSE returns the propensity score mean-squared error obtained from an original and synthetic dataset. For each variable, numeric values in the synthetic data are replaced with the nearest value in the original data. This prevents tree-based classifiers (used to estimate the propensity scores) to produce undesirable splits based only on the marginal distributions.

Usage

```
pMSE(orig, synth, method = c("cart", "gbm", "glm"), formula = syn ~ ., ...)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.
method	One of "cart", "gbm" and "glm". Specifies the method to be used to estimate the propensity scores.
formula	A formula, used to specify the covariates and interaction terms to be used to estimate the propensity scores. The dependent variable is named "syn".
...	Additional arguments to be passed to the function used to estimate the propensity scores.

Value

The metric value.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

pMSE(orig, synth, method = "glm")
```

---

pMSE_cp *Heuristics for the complexity parameter of tree-based estimation of propensity scores*

Description

pMSE_cp returns the mean complexity parameter obtained by tuning decision trees (with rpart::rpart()) to estimate propensity scores from a list of synthetic dataframes. Typically, this would be used to choose the complexity parameter for computing pMSE values.

Usage

```
pMSE_cp(orig, synth_l, xval = 25, ...)
```

Arguments

orig	A dataframe containing original data.
synth_l	A list of dataframes containing synthetic data, each with the same variables as orig.
xval	Number of cross-validations to perform.
...	Additional arguments to be passed to the rpart::rpart() function.

Value

The mean complexity value.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

cp <- pMSE_cp(orig, list(synth))

pMSE(orig, synth, method = "cart", cp = cp)
```

---

predict.pgb *Predict method for PGB models*

Description

predict.pgb predicts quantile values from a trained PGB model.

Usage

```
## S3 method for class 'pgb'
predict(object, newdata, project = TRUE, ...)
```

Arguments

object	An object of class "pgb".
newdata	A dataframe, structured like the training data (with or without the response variable).
project	A boolean, whether to correct for quantile crossings with an isotonic regression.
...	Further arguments passed to or from other methods.

Value

A matrix of predictions, where each line corresponds to an observation and each column to a quantile level.

Examples

```
data(iris)

fit <- pgb_cvh(Sepal.Width ~ Petal.Length, data = iris)

preds <- predict(fit, iris)

plot(iris$Petal.Length, iris$Sepal.Width)

matlines(iris$Petal.Length[order(iris$Petal.Length)],
         preds[order(iris$Petal.Length), ], type = "l", lty = 1)
```

---

predict_cde_pgb *Predict conditional densities from PGB models*

Description

`predict_cde_pgb` predicts smooth conditional densities from a trained PGB model.

Usage

```
predict_cde_pgb(object, newdata, y_grid)
```

Arguments

`object`	An object of class `"pgb"`.
`newdata`	A dataframe, structured like the training data (with or without the response variable).
`y_grid`	A vector of increasing numeric variables, the grid on which the densities will be evaluated.

Value

A matrix of predictions, where each line corresponds to an observation and each column to a density value on the provided grid.

Examples

```
data(iris)

fit <- pgb_cvh(Sepal.Width ~ Sepal.Length, data = iris)

y_grid <- seq(from = min(iris$Sepal.Width), to = max(iris$Sepal.Width), length.out = 200)
ids_x <- c(25, 80, 125)

cde <- predict_cde_pgb(fit, iris[ids_x, ], y_grid = y_grid)

x0 <- iris[ids_x, "Sepal.Length"]
s <- diff(range(iris$Sepal.Length)) / 8 / max(cde)

plot(iris$Sepal.Length, iris$Sepal.Width, xlab = "Sepal.Length", ylab = "Sepal.Width")
for(i in seq_len(nrow(cde))) lines(x0[i] + s * cde[i, ], y_grid, lwd = 2)
```

---

predict_cde_pgb_raw *Piecewise-constant conditional densities from PGB models*

Description

predict_cde_pgb_raw predicts histogram-like conditional densities (i.e. with no smoothing performed) from a trained PGB model.

Usage

```
predict_cde_pgb_raw(object, newdata, y_grid)
```

Arguments

object	An object of class "pgb".
newdata	A dataframe, structured like the training data (with or without the response variable).
y_grid	A vector of increasing numeric variables, the grid on which the densities will be evaluated.

Value

A matrix of predictions, where each line corresponds to an observation and each column to a density value on the provided grid.

Examples

```
data(iris)

fit <- pgb_cvh(Sepal.Width ~ Sepal.Length, data = iris, select_h = "none")

y_grid <- seq(from = min(iris$Sepal.Width), to = max(iris$Sepal.Width), length.out = 200)
ids_x <- c(25, 80, 125)

cde <- predict_cde_pgb_raw(fit, iris[ids_x, ], y_grid = y_grid)

x0 <- iris[ids_x, "Sepal.Length"]
s <- diff(range(iris$Sepal.Length)) / 8 / max(cde)

plot(iris$Sepal.Length, iris$Sepal.Width, xlab = "Sepal.Length", ylab = "Sepal.Width")
for(i in seq_len(nrow(cde))) lines(x0[i] + s * cde[i, ], y_grid, lwd = 2)
```

---

resample *Sample conditionally on vector length*

Description

`resample` samples from a vector if it contains more than one element, and otherwise returns its only element.

Usage

```
resample(vec, ...)
```

Arguments

`vec`	A vector.
`...`	Additional arguments passed to `base::sample()`.

Value

A sampled value or vector, depending on `...`.

See Also

`base::sample()`

Examples

```
vec <- 1:5 # sample and resample coincide when given a vector of length > 1

set.seed(1234)
sample(vec, 1)

set.seed(1234)
resample(vec, 1)

vec <- c(5) # with vectors of length 1, sample can return values that are not in the vector

set.seed(1234)
sample(vec, 1) # sample(c(5), 1) is equivalent to sample(1:5, 1)

set.seed(1234)
resample(vec, 1) # no problem with resample
```

---

round_synth *Round synthetic numeric variables to the precision observed in the original data*

Description

Round synthetic numeric variables to the precision observed in the original data

Usage

```
round_synth(orig, synth)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.

Value

A dataframe whose numeric columns are rounded with the precision observed in orig.

Examples

```
data(iris)

orig <- iris[sapply(iris, is.numeric)]

# synthesis by sampling from a multivariate Gaussian distribution
# with parameters estimated from the data
synth <- setNames(
  as.data.frame(
```

```
    sweep(
      matrix(rnorm(150 * ncol(orig)), 150) %*%
      chol(cov(orig) * (nrow(orig) - 1) / nrow(orig)),
      2,
      colMeans(orig),
      "+"
    )
  ),
  names(orig)
)

sapply(synth, get_prec)

synth <- round_synth(orig, synth)

sapply(synth, get_prec)
```

---

tsAUC *ts-AUC*

Description

tsAUC computes the AUC and associated p value from a two-sample test according to doi:10.1145/3411408.3411422.

Usage

```
tsAUC(orig, synth, ntreeTry = 50, ntree = 500, ...)
```

Arguments

orig	A dataframe containing original data.
synth	A dataframe containing synthetic data, with the same variables as orig.
ntreeTry	An integer, the number of trees used at the tuning step.
ntree	An integer, the number of trees used at the training step.
...	options to be given to randomForest::randomForest().

Value

A list with the following elements:

auc The estimated area under the ROC curve.

p.value The p value associated with a one-sided Wilcoxon test.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

tsAUC(orig, synth)$p.value
```

`univ_att_prob` *Univariate correct attribution probability*

Description

`univ_att_prob` computes the univariate correct attribution probability from given parameters, as described in doi:10.48550/arXiv.2310.06571.

Usage

```
univ_att_prob(row, data, rad)
```

Arguments

`row`	The target row of the dataframe.
`data`	A dataframe.
`rad`	A vector containing radii, one for each numeric variable of the dataframe. The order of the radii must match the order of the variables in the dataframe.

Details

When the input dataframe contains only categorical variables, the value returned is the proportion of rows in the dataframe having the same combination of values as the input row. When the input dataframe also contains numeric variables, this value is weighted according to the proximity between these variables in the input row vs. each other row of the dataframe. The weights are given by radii passed as parameters.

Value

The computed probability.

Examples

```
data(iris)

orig <- iris

# simple synthesis by sampling from the product of the marginal distributions
synth <- as.data.frame(lapply(orig, function(x) x[sample(length(x), nrow(orig), replace = TRUE)]))

target_row <- orig[1, ]

radii <- sapply(Filter(is.numeric, orig), function(x) {
  d <- diff(sort(x))
  mean(pmin(c(Inf, d), c(d, Inf)))
})

univ_att_prob(target_row, synth, radii)
```

Index