Package 'CCI'

March 3, 2026

Type Package

Title Computational Test for Conditional Independence

Version 0.3.6.1

Date 2026-03-03

Maintainer Christian Thorjussen <christianbern@gmail.com>

Description Tool for performing computational testing for conditional independence between variables in a dataset. 'CCI' implements permutation in combination with Monte Carlo Cross-Validation in generating null distributions and test statistics. For more details see Computational Test for Conditional Independence (2024) <doi:10.3390/a17080323>.

Imports ggplot2, dplyr, caret, xgboost, ranger, stats, data.table, e1071, rlang, progress, kknn

Suggests testthat, knitr, rmarkdown

License GPL (>= 2)

URL https://github.com/khliland/CCI

BugReports https://github.com/khliland/CCI/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

Author Christian Thorjussen [aut, cre] (ORCID: <https://orcid.org/0009-0005-5006-6491>), Kristian Hovde Liland [aut] (ORCID: <https://orcid.org/0000-0001-6468-9423>)

Repository CRAN

Date/Publication 2026-03-03 13:50:02 UTC

Contents

`add_interaction_terms` *Creates interaction terms for specified variables in a data frame Interaction terms are named as* `<var1>_int_<var2>` *(e.g.,* `Z1_int_Z2` *for the product of* `Z1` *and* `Z2`*).*

Description

Creates interaction terms for specified variables in a data frame Interaction terms are named as `<var1>_int_<var2>` (e.g., `Z1_int_Z2` for the product of `Z1` and `Z2`).

Usage

```
add_interaction_terms(data, Z, mode = c("numeric_only", "mixed"))
```

Arguments

data	Data frame. The data frame containing the variables for which interaction terms are to be created.
Z	Character vector. The names of the variables for which interaction terms are to be created.
mode	Character. Specifies the type of interaction terms to create. Options are: numeric_only (only numeric-numeric interactions as products) or mixed (numeric-numeric as products, factor/character involved as categorical interactions). Default is "numeric_only".

Value

A list with two components:

- data: The modified data frame with added interaction terms.
- new_terms: A character vector of the names of the added interaction terms (e.g., Z1_int_2).

Examples

```
data_generator <-  function(N){
Z1 <- rnorm(N,0,1)
Z2 <- rnorm(N,0,1)
X <- rnorm(N, Z1 + Z2, 1)
Y <- rnorm(N, Z1 + Z2, 1)
df <- data.frame(Z1, Z2, X, Y)
return(df)
}
dat <- data_generator(250)
interaction_terms <- add_interaction_terms(data = dat, Z = c("Z1", "Z2"))
head(interaction_terms$data$Z1_int_Z2)
```

---

add_poly_terms *Creates polynomial terms for specified variables in a data frame Polynomial terms are named as* <variable>_d_<degree> *(e.g.,* Z1_d_2 *for the square of* Z1*).*

Description

Creates polynomial terms for specified variables in a data frame Polynomial terms are named as <variable>_d_<degree> (e.g., Z1_d_2 for the square of Z1).

Usage

```
add_poly_terms(data, Z, degree = 3, poly = TRUE)
```

Arguments

data	Data frame. The data frame containing the variables for which polynomial terms are to be created.
Z	Character vector. The names of the variables for which polynomial terms are to be created.
degree	Integer. The maximum degree of polynomial terms to be created. Default is 3.
poly	Logical. If TRUE, polynomial terms will be created. If FALSE, no polynomial terms will be created. Default is TRUE.

Value

A list with two components:

- data: The modified data frame with added polynomial terms.
- new_terms: A character vector of the names of the added polynomial terms (e.g., Z1_d_2).

#'

Examples

```
set.seed(123)
data_generator <-  function(N){
Z1 <- rnorm(N,0,1)
Z2 <- rnorm(N,0,1)
X <- rnorm(N, Z1 + Z2, 1)
Y <- rnorm(N, Z1 + Z2, 1)
df <- data.frame(Z1, Z2, X, Y)
return(df)
}
dat <- data_generator(250)
poly_terms <- add_poly_terms(data = dat, Z = c("Z1", "Z2"), degree = 3, poly = TRUE)
print(poly_terms$new_terms)
```

---

build_formula *Build an expanded formula with poly and interaction terms*

Description

Build an expanded formula with poly and interaction terms

Usage

```
build_formula(formula, poly_terms = NULL, interaction_terms = NULL)
```

Arguments

`formula`	A base formula in the format Y ~ X | Z1 + Z2
`poly_terms`	Character vector of polynomial term names
`interaction_terms`	Character vector of interaction term names

Value

A formula object combining all terms

Examples

```
poly_terms <- c("Z1_d_2", "Z2_d_2")
interaction_terms <- c("Z1_int_Z2")
formula <- Y ~ X | Z1 + Z2
final_formula <- build_formula(formula, poly_terms, interaction_terms)
print(final_formula)
```

`CCI.direction` *Choose Direction for testing for the CCI test*

Description

This function selects the best direction for the CCI test based on cross validation. For the condition Y || X | Z, the function return the recommended formula either Y ~ X | Z or X ~ Y | Z .

Usage

```
CCI.direction(
  formula,
  data,
  method = "rf",
  folds = 4,
  nrounds = 600,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1,
  poly = TRUE,
  degree = 3,
  interaction = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

formula	A formula object specifying the model to be fitted.
data	A data frame containing the variables specified in the formula.
method	A character string specifying the method to be used for model fitting. Options include "rf" (random forest), "xgboost" (XGBoost), "nnet" (neural network), "gpr" (Gaussian process regression), and "svm" (support vector machine).
folds	An integer specifying the number of folds for cross-validation. Default is 4.
nrounds	Integer. The number of rounds (trees) for methods like xgboost, ranger, and lightgbm. Default is 600.
max_depth	Integer. The maximum depth of the trees for methods like xgboost. Default is 6.
eta	Numeric. The learning rate for methods like xgboost. Default is 0.3.
gamma	Numeric. The minimum loss reduction required to make a further partition on a leaf node of the tree for methods like xgboost. Default is 0.
colsample_bytree	Numeric. The subsample ratio of columns when constructing each tree for methods like xgboost. Default is 1.
min_child_weight	Numeric. The minimum sum of instance weight (hessian) needed in a child for methods like xgboost. Default is 1.
subsample	Numeric. The proportion of the data to be used for subsampling. Default is 1 (no subsampling).
poly	Logical. If TRUE, polynomial terms of the conditioning variables are included in the model. Default is TRUE.
degree	Integer. The degree of polynomial terms to include if poly is TRUE. Default is 3.
interaction	Logical. If TRUE, interaction terms of the conditioning variables are included in the model. Default is TRUE.
verbose	Logical. If TRUE, prints additional information during the execution. Default is FALSE.
...	Additional arguments to be passed to the model fitting function.

Value

A formula object specifying the selected model direction.

`CCI.pretuner` *CCI tuner function for CCI test*

Description

The `CCI.tuner` function performs a grid search over parameters for a conditional independence test using machine learning model supported by CCI.test. The tuner use the caret package for tuning.

Usage

```
CCI.pretuner(
  formula,
  data,
  method = "rf",
  metric = "RMSE",
  validation_method = "cv",
  folds = 4,
  training_share = 0.7,
  tune_length = 4,
  random_grid = TRUE,
  samples = 35,
  poly = TRUE,
  degree = 3,
  interaction = TRUE,
  verboseIter = FALSE,
  include_explanatory = FALSE,
  verbose = FALSE,
  parallel = FALSE,
  mtry = 1:10,
  nrounds = c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000),
  eta = seq(0.01, 0.3, by = 0.05),
  max_depth = 2:6,
  gamma = c(0, 1, 2, 3),
  colsample_bytree = c(0.8, 0.9, 1),
  min_child_weight = c(1, 3),
  subsample = 1,
  sigma = seq(0.1, 2, by = 0.3),
  C = seq(0.1, 2, by = 0.5),
  ...
)
```

Arguments

`formula`	Model formula specifying the relationship between dependent and independent variables.
`data`	A data frame containing the variables specified in the formula.

`method` Character. Specifies the machine learning method to use. Supported methods are random forest "rf", extreme gradient boosting "xgboost" and Support Vector Machine "svm".

`metric` Character. The performance metric to optimize during tuning. Default is "RMSE".

`validation_method` Character. Specifies the resampling method. Default is "cv".

`folds` Integer. The number of folds for cross-validation during the tuning process. Default is 10.

`training_share` Numeric. For leave-group out cross-validation: the training percentage. Default is 0.7.

`tune_length` Integer. The number of parameter combinations to try during the tuning process. Default is 10.

`random_grid` Logical. If TRUE, a random grid search is performed. If FALSE, a full grid search is performed. Default is TRUE.

`samples` Integer. The number of random samples to take from the grid. Default is 30.

`poly` Logical. If TRUE, polynomial terms of the conditional variables are included in the model. Default is TRUE.

`degree` Integer. The degree of polynomial terms to include if poly is TRUE. Default is 3.

`interaction` Logical. If TRUE, interaction terms of the conditional variables are included in the model. Default is TRUE.

`verboseIter` Logical. If TRUE, the function will print the tuning process. Default is FALSE.

`include_explanatory` Logical. If TRUE, given the condition Y || X | Z, the function will include explanatory variable X in the model for Y. Default is FALSE

`verbose` Logical. If TRUE, the function will print the tuning process. Default is FALSE..

`parallel` Logical. If TRUE, the function will use parallel processing. Default is TRUE.

`mtry` Integer. The number of variables randomly sampled as candidates at each split for random forest. Default is 1:5.

`nrounds` Integer. The number of rounds (trees) for methods such as xgboost and random forest. Default is seq(50, 200, by = 25).

`eta` Numeric. The learning rate for xgboost. Default is seq(0.01, 0.3, by = 0.05).

`max_depth` Integer. The maximum depth of the tree for xgboost. Default is 1:6.

`gamma` Numeric. The minimum loss reduction required to make a further partition on a leaf node for xgboost. Default is seq(0, 5, by = 1).

`colsample_bytree` Numeric. The subsample ratio of columns when constructing each tree for xgboost. Default is seq(0.5, 1, by = 0.1).

`min_child_weight` Integer. The minimum sum of instance weight (hessian) needed in a child for xgboost. Default is 1:5.

`subsample` Numeric. The subsample ratio of the training. Default is 1.

sigma	Numeric. The standard deviation of the Gaussian kernel for Gaussian Process Regression. Default is seq(0.1, 2, by = 0.3).
C	Numeric. The regularization parameter for Support Vector Machine. Default is seq(0.1, 2, by = 0.5).
...	Additional arguments to pass to the CCI.tuner function.

Value

A list containing:

- best_param: A data frame with the best parameters.
- tuning_result: A data frame with all tested parameter combinations and their performance metrics.
- warnings: A character vector of warnings issued during tuning.

See Also

CCI.test perm.test, print.summary.CCI, plot.CCI, QQplot

Examples

```
set.seed(123)
data <- data.frame(x1 = rnorm(100), x2 = rnorm(100), x3 = rnorm(100), y = rnorm(100))
# Tune random forest parameters
result <- CCI.pretuner(formula = y ~ x1 | x2 + x3,
data = data,
samples = 5,
folds = 3,
method = "rf")
```

CCI.test *Computational test for conditional independence based on ML and Monte Carlo Cross Validation*

Description

The CCI.test function performs a conditional independence test using a specified machine learning model or a custom model provided by the user. It calculates the test statistic, generates a null distribution via permutations, computes p-values, and optionally generates a plot of the null distribution with the observed test statistic. The 'CCI.test' function serves as a wrapper around the 'perm.test' function

Usage

```
CCI.test(
  formula = NULL,
  data,
  p = 0.5,
  nperm = 160,
  nrounds = 600,
  mtry = NULL,
  metric = "Auto",
  method = "rf",
  choose_direction = FALSE,
  parametric = FALSE,
  poly = TRUE,
  degree = 3,
  robust = TRUE,
  subsample = "Auto",
  subsample_set,
  min_child_weight = 1,
  colsample_bytree = 1,
  eta = 0.3,
  gamma = 0,
  max_depth = 6,
  interaction = TRUE,
  mode = "numeric_only",
  metricfunc = NULL,
  mlfunc = NULL,
  tail = NA,
  tune = FALSE,
  samples = 35,
  folds = 5,
  tune_length = 10,
  k = 15,
  center = TRUE,
  scale = TRUE,
  eps = 1e-15,
  positive = NULL,
  kernel = "optimal",
  distance = 2,
  seed = NA,
  random_grid = TRUE,
  nthread = 2,
  verbose = FALSE,
  progress = TRUE,
  ...
)
```

Arguments

formula	Model formula specifying the relationship between dependent and independent variables. (Ex: Y ~ X | Z1 + Z2 for Y || X | Z1, Z2)
data	A data frame containing the variables specified in the formula.
p	Numeric. Proportion of data used for training the model. Default is 0.5.
nperm	Integer. The number of permutations to perform. Default is 60.
nrounds	Integer. The number of rounds (trees) for methods 'xgboost' and 'rf' Default is 600.
mtry	Number of variables to possibly split at in each node for method 'rf'. Default is NULL (sqrt of number of variables).
metric	Character. Specifies the type of data: "Auto", "RMSE" or "Kappa". Default is "Auto".
method	Character. Specifies the machine learning method to use. Supported methods are random forest "rf", extreme gradient boosting "xgboost", support vector machine 'svm' and K-nearest neighbour 'KNN'. Default is "rf".
choose_direction	Logical. If TRUE, the function will choose the best direction for testing. Default is FALSE.
parametric	Logical, indicating whether to compute a parametric p-value instead of the empirical p-value. A parametric p-value assumes that the null distribution is gaussian. Default is FALSE.
poly	Logical. If TRUE, polynomial terms of the conditional variables are included in the model. Default is TRUE.
degree	Integer. The degree of polynomial terms to include if poly is TRUE. Default is 3.
robust	Logical. If TRUE, uses a robust method for permutation. Default is TRUE.
subsample	Character. Specifies whether to use automatic subsampling based on sample size ("Auto"), user-defined subsampling ("Yes"), or no subsampling ("No"). Default is "Auto"
subsample_set	Numeric. If subsample is set to "Yes", this parameter defines the proportion of data to use for subsampling. Default is NA.
min_child_weight	Numeric. The minimum sum of instance weight (hessian) needed in a child for methods like xgboost. Default is 1.
colsample_bytree	Numeric. The subsample ratio of columns when constructing each tree for methods like xgboost. Default is 1.
eta	Numeric. The learning rate for methods like xgboost. Default is 0.3.
gamma	Numeric. The minimum loss reduction required to make a further partition on a leaf node of the tree for methods like xgboost. Default is 0.
max_depth	Integer. The maximum depth of the trees for methods like xgboost. Default is 6.
interaction	Logical. If TRUE, interaction terms of the conditional variables are included in the model. Default is TRUE.

mode	Character. Specifies the mode of operation: "numeric_only" or "mixed". Default is "numeric_only".
metricfunc	Optional the user can pass a custom function for calculating a performance metric based on the model's predictions. Default is NULL.
mlfunc	Optional the user can pass a custom machine learning wrapper function to use instead of the predefined methods. Default is NULL.
tail	Character. Specifies whether to calculate left-tailed or right-tailed p-values, depending on the performance metric used. Only applicable if using metricfunc or mlfunc. Default is NA.
tune	Logical. If TRUE, the function will perform hyperparameter tuning for the specified machine learning method. Default is FALSE.
samples	Integer. Number of hyperparameter combinations used in tuning. Default is 35.
folds	Integer. The number of folds for cross-validation during the tuning process. Default is 5.
tune_length	Integer. The number of parameter combinations to try during the tuning process. Default is 10.
k	Integer. The number of nearest neighbors to use for KNN method. Default is 15.
center	Logical. If TRUE, the data will be centered before fitting the model
scale	Logical. If TRUE, the data will be scaled before fitting the model. Default is TRUE.
eps	Numeric. A small value to avoid division by zero in some calculations.
positive	Character. The name of the positive class (KNN) in the data, used for classification tasks. Default is NULL.
kernel	Character. The kernel type to use for KNN method. Default is "optimal".
distance	Numeric. Parameter of Minkowski distance for the "KNN" method. Default is 2.
seed	Integer. Set the seed for reproducing results. Default is NA.
random_grid	Logical. If TRUE, a random grid search is performed. If FALSE, a full grid search is performed. Default is TRUE.
nthread	Integer. The number of threads to use for parallel processing. Default is 1.
verbose	Logical. If TRUE, additional information is printed during the execution of the function. Default is FALSE.
progress	Logical. If TRUE, a progress bar is displayed during the permutation process. Default is TRUE.
...	Additional arguments to pass to the perm.test function.

Value

Invisibly returns the result of perm.test, which is an object of class 'CCI' containing the null distribution, observed test statistic, p-values, the machine learning model used, and the data.

See Also

perm.test, print.summary.CCI, plot.CCI, CCI.pretuner, QQplot

Examples

```
set.seed(123)
data <- data.frame(x1 = stats::rnorm(100), x2 = stats::rnorm(100), y = stats::rnorm(100))
result <- CCI.test(y ~ x1 | x2, data = data, nperm = 25, interaction = FALSE)
summary(result)
```

---

check_formula *Check the formula statement*

Description

This function verifies that all variables specified in the formula are present in the provided data frame. If any variables are missing, the function will stop and return an error message listing the missing variables.

Usage

```
check_formula(formula, data)
```

Arguments

formula	Formula. The model formula that specifies the relationship between the dependent and independent variables.
data	Data frame. The data frame in which to check for the presence of variables specified in the formula.

Value

Invisibly returns NULL if all variables are present. Stops with an error if any variables are missing.

---

clean_formula *Clean and Reformat Formula String*

Description

This function processes and reformats formula string to ensure it is in the correct format for conditional independence testing. The function checks if the formula uses the '+' operator for additive models and transforms it into a format that includes a conditioning variable separated by '|'.

Usage

```
clean_formula(formula)
```

Arguments

`formula`	Formula. The model formula that specifies the relationship between the dependent and independent variables, and potentially the conditioning variables. The formula is expected to follow the format `Y ~ X + Z1 + Z2` or `Y ~ X | Z1 + Z2`.

Value

A reformatted formula in the correct format for conditional independence testing. The returned formula will either retain the original format or be transformed to include conditioning variables.

Examples

```
clean_formula(y ~ x | z + v)
clean_formula(y ~ x + z + v)
```

`ExponentialNoise` *Example dataset: ExponentialNoise*

Description

A dataset containing simulated conditional independence test results.

Usage

```
ExponentialNoise
```

Format

A data frame with 600 rows and 3 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

get_pvalues *P-value Calculation Based on Null Distribution and Test Statistic*

Description

This function calculates p-values based on the comparison of a test statistic against a null distribution. It can perform either empirical or parametric p-value calculations and supports both left-tailed and right-tailed tests.

Usage

```
get_pvalues(
  dist,
  test_statistic,
  parametric = FALSE,
  tail = c("left", "right")
)
```

Arguments

dist	Numeric vector. Represents the null distribution of the test statistic.
test_statistic	Numeric. The observed test statistic for which the p-value is to be calculated.
parametric	Logical. If TRUE, calculates parametric p-values assuming the null distribution is normal. If FALSE, calculates empirical p-values. Default is FALSE.
tail	Character. Specifies whether to calculate left-tailed or right-tailed p-values. Must be either "left" or "right". Default is "left".

Value

Numeric. The calculated p-value.

Examples

```
set.seed(123)
null_dist <- rnorm(1000)
observed_stat <- 1.5
p_value <- get_pvalues(null_dist, observed_stat, parametric = FALSE, tail = "right")
print(p_value)
```

get_tuned_params *Get the best parameters after tuning with CCI.tuner*

Description

Get the best parameters after tuning with CCI.tuner

Usage

```
get_tuned_params(tuned_model)
```

Arguments

tuned_model	A model object returned from the CCI.pretuner function. This object contains the tuned parameters and other relevant information.

Value

A named list of tuned parameters specific to the model method (e.g., `mtry` for random forest, `eta`, `max_depth` for xgboost). Returns `NULL` for unsupported methods.

HardCase *Example dataset: HardCase*

Description

A dataset containing simulated conditional independence test results.

Usage

```
HardCase
```

Format

A data frame with 500 rows and 3 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

is_categorical_Z_any *Check whether Z contains at least one categorical variable*

Description

Categorical is defined as factor (and optionally character and logical). This helper is intentionally strict about inputs and does not modify data.

Usage

```
is_categorical_Z_any(sub_data, Z, allow_character = TRUE, allow_logical = TRUE)
```

Arguments

sub_data	data.frame containing the Z columns.
Z	character vector of column names defining the conditioning set.
allow_character	logical; treat character as categorical. Default TRUE.
allow_logical	logical; treat logical as categorical. Default TRUE.

Value

logical scalar.

make_strata_from_categorical_Z *Create strata from the categorical subset of Z*

Description

Uses interaction() on the categorical Z columns only. Characters (and logicals, if enabled) are coerced to factor for stable behavior.

Usage

```
make_strata_from_categorical_Z(
  sub_data,
  Z,
  allow_character = TRUE,
  allow_logical = TRUE
)
```

Arguments

`sub_data`	data.frame containing Z columns.
`Z`	character vector of Z column names.
`allow_character`	logical; treat character as categorical. Default TRUE.
`allow_logical`	logical; treat logical as categorical. Default TRUE.

Details

Rows with NA in any stratification column will receive NA strata.

Value

A factor defining strata (same length as nrow(sub_data)).

`NonLinearCategorization`

Example dataset: NonLinearCategorization

Description

A dataset containing simulated data from a non-linear transformation followed by categorization.

Usage

```
NonLinearCategorization
```

Format

A data frame with 600 rows and 3 variables:

X Numeric vector

Y Numeric vector

Z Conditioning variable

Source

Simulated data.

NonLinNormal *Example dataset: NonLinNormal*

Description

A dataset containing simulated data from a non-linear transformation of a multivariate normal distribution.

Usage

```
NonLinNormal
```

Format

A data frame with 500 rows and 4 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

NonLinNormalZs_d0 *Example dataset: NonLinNormalZs_d0*

Description

A dataset containing simulated data with uniform noise.

Usage

```
NonLinNormalZs_d0
```

Format

A data frame with 1000 rows and 15 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Z3 Conditioning variable

Z4 Conditioning variable

Z5 Conditioning variable

Z6 Conditioning variable

Z7 Conditioning variable

Z8 Conditioning variable

Z9 Conditioning variable

Z10 Conditioning variable

Z11 Conditioning variable

Z12 Conditioning variable

Z13 Conditioning variable

Z14 Conditioning variable

Z15 Conditioning variable

Source

Simulated data.

`NonLinNormalZs_d05` *Example dataset: NonLinNormalZs_d05*

Description

A dataset containing simulated data with uniform noise.

Usage

```
NonLinNormalZs_d05
```

Format

A data frame with 1000 rows and 15 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Z3 Conditioning variable

Z4 Conditioning variable

Z5 Conditioning variable

Z6 Conditioning variable

Z7 Conditioning variable

Z8 Conditioning variable

Z9 Conditioning variable

Z10 Conditioning variable

Z11 Conditioning variable

Z12 Conditioning variable

Z13 Conditioning variable

Z14 Conditioning variable

Z15 Conditioning variable

Source

Simulated data.

`NormalData` *Example dataset: NormalData*

Description

A dataset containing simulated data from a multivariate normal distribution.

Usage

```
NormalData
```

Format

A data frame with 400 rows and 4 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

@source Simulated data.

perm.test *Permutation Test for Conditional Independence*

Description

Permutation Test for Conditional Independence

Usage

```
perm.test(
  formula,
  data,
  p = 0.5,
  nperm = 160,
  subsample = 1,
  metric = "RMSE",
  method = "rf",
  nrounds = 600,
  mtry = NULL,
  parametric = FALSE,
  tail = NA,
  robust = TRUE,
  metricfunc = NULL,
  mlfunc = NULL,
  nthread = 1,
  progress = TRUE,
  k = 15,
  center = TRUE,
  scale = TRUE,
  eps = 1e-15,
  positive = NULL,
  kernel = "optimal",
  distance = 2,
  ...
)
```

Arguments

formula	Model formula or DAGitty object specifying the relationship between dependent and independent variables.
data	A data frame containing the variables specified in the formula.
p	Proportion of data to use for training the model. Default is 0.5.
nperm	Number of permutations to perform. Default is 160.
subsample	The proportion of the data to be used. Default is 1 (no subsampling).
metric	Type of metric: "RMSE", "Kappa" or "LogLoss". Default is 'RMSE'.

method	The machine learning method to use for the learner. Supported methods include "rf", "xgboost", "KNN" and "svm". Default is "rf".
nrounds	Number of rounds (trees) for methods 'xgboost' and 'rf'. Default is 600.
mtry	Number of variables to possibly split at in each node for method 'rf'. Default is NULL (sqrt of number of variables).
parametric	Logical. If TRUE, a parametric p-value is calculated instead of an empirical p-value. Default is FALSE.
tail	Specifies whether the test is one-tailed ("left" or "right") or two-tailed. Default is NA.
robust	Logical. If TRUE, uses a robust method for permutation. Default is TRUE.
metricfunc	An optional custom function to calculate the performance metric based on the model's predictions. Default is NULL.
mlfunc	An optional custom machine learning function to use instead of the predefined methods. Default is NULL.
nthread	Integer. The number of threads to use for parallel processing for method 'rf' and 'xgboost'. Default is 1.
progress	Logical. If TRUE, a progress bar is displayed during the permutation process. Default is TRUE.
k	Integer. The number of nearest neighbors for the "KNN" method. Default is 15.
center	Logical. If TRUE, the data is centered before model fitting. Default is TRUE.
scale	Logical. If TRUE, the data is scaled before model fitting. Default is TRUE.
eps	Numeric. A small value added to avoid division by zero. Default is 1e-15.
positive	Character vector. Specifies which levels of a factor variable should be treated as positive class in classification tasks. Default is NULL.
kernel	Character string specifying the kernel type for method option "KNN" . Possible choices are "rectangular" (which is standard unweighted knn), "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv", "gaussian" and "optimal". Default is "optimal".
distance	Numeric. Parameter of Minkowski distance for the "KNN" method. Default is 2.
...	Additional arguments to pass to the machine learning model fitting function.

Value

An object of class 'CCI' containing the null distribution, observed test statistic, p-values, the machine learning model used, and the data.

See Also

print.CCI, summary.CCI, plot.CCI, QQplot

Examples

```
set.seed(123)
dat <- data.frame(x1 = rnorm(100),
x2 = rnorm(100),
x3 = rnorm(100),
x4 = rnorm(100),
y = rnorm(100))
perm.test(y ~ x1 | x2 + x3 + x4, data = dat, nperm = 25)
```

`permute_within_strata` *Stratified permutation of x within strata*

Description

Permutes values of 'x' within each stratum. By default, rows with NA in 'x' or 'strata' are left unchanged to preserve alignment.

Usage

```
permute_within_strata(x, strata, seed = NULL, na_action = c("keep", "drop"))
```

Arguments

`x`	vector to permute.
`strata`	factor-like vector defining strata.
`seed`	optional seed for reproducibility.
`na_action`	how to handle NA rows: "keep" (default) keeps them fixed, "drop" removes them (returns shorter vector).

Value

x_star permuted within strata.

Examples

```
set.seed(123)
x <- 1:10
strata <- rep(letters[1:2], each = 5)
x_permuted <- permute_within_strata(x, strata, seed = 123)
```

plot.CCI *Plot for CCI testing*

Description

Plot for CCI testing

Usage

```
## S3 method for class 'CCI'
plot(
  x,
  fill_color = "lightblue",
  title.size = 14,
  axis.text.x = 13,
  axis.text.y = 13,
  strip.text.x = 13,
  strip.text.y = 13,
  legend.text = 13,
  legend.title = 13,
  axis.title.x = 13,
  axis.title.y = 13,
  base_size = 13,
  ...
)
```

Arguments

x	Object of class 'CCI'
fill_color	Color for the histogram fill
title.size	Size of the plot title
axis.text.x	Size of x-axis text
axis.text.y	Size of y-axis text
strip.text.x	Size of x-axis strip text
strip.text.y	Size of y-axis strip text
legend.text	Size of legend text
legend.title	Size of legend title
axis.title.x	Size of x-axis title
axis.title.y	Size of y-axis title
base_size	Base font size
...	Additional arguments to ggplot2

Value

A plot of the null distribution and the test statistic in ggplot2 format.

See Also

print.CCI, summary.CCI, plot.CCI, perm.test

Examples

```
dat <- data.frame(x1 = rnorm(100), x2 = rnorm(100), y = rnorm(100))
cci <- CCI.test(y ~ x1 + x2, data = dat, interaction = FALSE, nperm = 30)
plot(cci)
```

PoissonNoise *Example dataset: PoissonNoise*

Description

A dataset containing simulated data from a Poisson distribution.

Usage

```
PoissonNoise
```

Format

A data frame with 1000 rows and 4 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

PolyData *Example dataset: PolyData*

Description

A dataset containing simulated data from a polynomial relationship.

Usage

```
PolyData
```

Format

A data frame with 600 rows and 4 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

print.summary.CCI *Print and summary methods for the CCI class*

Description

Print and summary methods for the CCI class

Usage

```
## S3 method for class 'summary.CCI'
print(x, ...)
```

Arguments

x	Object of class 'CCI'
...	Additional arguments to print/summary

Value

The print methods have no return value, the summary methods return an object of class 'summary.CCI'.

See Also

perm.test, plot.CCI, QQplot

QQplot *QQ-plot for multiple testing in CCI*

Description

QQ-plot for multiple testing in CCI

Usage

```
QQplot(
  object,
  title.size = 14,
  axis.text.x = 13,
  axis.text.y = 13,
  strip.text.x = 13,
  strip.text.y = 13,
  legend.text = 13,
  legend.title = 13,
  axis.title.x = 13,
  axis.title.y = 13,
  progress = TRUE,
  ...
)
```

Arguments

object	Object of class 'CCI'
title.size	Size of the plot title
axis.text.x	Size of x-axis text
axis.text.y	Size of y-axis text
strip.text.x	Size of x-axis strip text
strip.text.y	Size of y-axis strip text
legend.text	Size of legend text
legend.title	Size of legend title
axis.title.x	Size of x-axis title
axis.title.y	Size of y-axis title
progress	Logical indicating whether to show progress during computation
...	Additional arguments to pass to the test.gen function.

Value

A QQ-plot of the p-values in ggplot2 format.

See Also

print.CCI, summary.CCI, plot.CCI, perm.test

Examples

```
dat <- data.frame(x1 = rnorm(100), x2 = rnorm(100), y = rnorm(100))
cci <- CCI.test(y ~ x1 | x2,
data = dat,
nperm = 25,
interaction = FALSE)
QQplot(cci)
```

---

test.gen *Generate the Test Statistic or Null Distribution Using Permutation*

Description

This function generates the test statistic or a null distribution through permutation for conditional independence testing. It supports various machine learning methods, including random forests, extreme gradient boosting, and allows for custom metric functions and model fitting functions.

Usage

```
test.gen(
  formula,
  data,
  method = "rf",
  metric = "RMSE",
  nperm = 160,
  subsample = 1,
  p = 0.5,
  nrounds = 600,
  mtry = NULL,
  nthread = 1,
  permutation = FALSE,
  robust = TRUE,
  metricfunc = NULL,
  mlfunc = NULL,
  progress = TRUE,
  center = TRUE,
  scale = TRUE,
  eps = 1e-15,
  k = 15,
```

```
  positive = NULL,
  kernel = "optimal",
  distance = 2,
  ...
)
```

Arguments

formula	Formula specifying the relationship between dependent and independent variables.
data	Data frame. The data containing the variables used.
method	Character. The modeling method to be used. Options include "xgboost" for gradient boosting, or "rf" for random forests or "svm" for Support Vector Machine.
metric	Character. The type of metric: can be "RMSE", "Kappa" or "LogLoss". Default is 'RMSE'
nperm	Integer. The number of generated Monte Carlo samples. Default is 160.
subsample	Numeric. The proportion of the data to be used for subsampling. Default is 1 (no subsampling).
p	Numeric. The proportion of the data to be used for training. The remaining data will be used for testing. Default is 0.5.
nrounds	Integer. The number of rounds (trees) for methods like 'xgboost' and 'rf'. Default is 600.
mtry	Integer. The number of variables to possibly split at in each node for method 'rf'. Default is the rounded down square root of numbers of columns in data.
nthread	Integer. The number of threads to use for parallel processing. Only relevant for methods 'rf' and 'xgboost'. Default is 1.
permutation	Logical. Whether to perform permutation of the 'X' variable. Used to generate a null distribution. Default is FALSE.
robust	Logical. If TRUE, automatically performs stratified permutation if all conditional variables are factor or categorical. Default is TRUE.
metricfunc	Function. A custom metric function provided by the user. It must take arguments: `actual`, `predictions`, and optionally `...`, and return a single numeric performance value.
mlfunc	Function. A custom machine learning function provided by the user. The function must have the arguments: `formula`, `data`, `train_indices`, `test_indices`, and `...`, and return a single value performance metric. Default is NULL.
progress	Logical. A logical value indicating whether to show a progress bar during when building the null distribution. Default is TRUE.
center	Logical. If TRUE, the data is centered before model fitting. Default is TRUE.
scale	Logical. If TRUE, the data is scaled before model fitting. Default is TRUE.
eps	Numeric. A small value added to avoid division by zero. Only relevant for method 'KNN'. Default is 1e-15.
k	Integer. The number of nearest neighbors for the "KNN" method. Default is 15.

positive	Character vector. Only relevant for method 'KNN'. Specifies which levels of a factor variable should be treated as positive class in classification tasks. Default is NULL.
kernel	Character. Only relevant for method 'KNN'. Specifies the kernel type for method option "KNN" . Possible choices are "rectangular" (which is standard unweighted knn), "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv", "gaussian" and "optimal". Default is "optimal".
distance	Numeric. Parameter of Minkowski distance for the "KNN" method. Default is 2.
...	Additional arguments to pass to the machine learning wrapper functions `wrapper_xgboost`, `wrapper_ranger`, `wrapper_knn` and `wrapper_svm`, or to a custom-built wrapper function.

Value

A list containing the test distribution.

Examples

```
set.seed(123)
data <- data.frame(x1 = rnorm(100),
x2 = rnorm(100),
x3 = rnorm(100),
x4 = rnorm(100),
y = rnorm(100))
result <- test.gen(formula = y ~ x1 | x2 + x3 + x4,
                   metric = "RMSE",
                   nperm = 50,
                   data = data)
hist(result$distribution)
```

---

unclean_formula *Convert CI-style formula Y ~ X | Z into regression-style Y ~ X + Z*

Description

Convert CI-style formula Y ~ X | Z into regression-style Y ~ X + Z

Usage

```
unclean_formula(f)
```

Arguments

f	A formula of the form Y ~ X | Z1 + Z2, or already Y ~ X + Z1 + Z2.

Value

A standard formula of the form Y ~ X + Z1 + Z2.

UniformNoise_large *Example dataset: UniformNoise_large*

Description

A dataset containing simulated data with uniform noise.

Usage

```
UniformNoise_large
```

Format

A data frame with 20000 rows and 4 variables:

X Numeric vector

Y Numeric vector

Z1 Conditioning variable

Z2 Conditioning variable

Source

Simulated data.

---

wrapper_knn *k-Nearest Neighbors (KNN) wrapper for CCI (kknn-based)*

Description

k-Nearest Neighbors (KNN) wrapper for CCI (kknn-based)

Usage

```
wrapper_knn(
  formula,
  data,
  train_indices,
  test_indices,
  metric,
  metricfunc = NULL,
  k = 15,
  eps = 1e-15,
```

```
  positive = NULL,
  kernel = "optimal",
  distance = 2,
  ...
)
```

Arguments

formula	Model formula
data	Data frame
train_indices	Indices for training rows
test_indices	Indices for test rows
metric	Performance metric: "RMSE" (regression), "Kappa" (classification), or "LogLoss" (classification)
metricfunc	Optional custom metric function: function(actual, predictions, ...)
k	Integer, number of neighbors (default 15)
eps	Small value to avoid log(0) in LogLoss calculations. Default is 1e-15.
positive	Character. The positive class label for binary classification (used in LogLoss). Default is NULL.
kernel	Character. Weighting kernel for kknn. Default "optimal".
distance	Numeric. Minkowski distance parameter. 2 = Euclidean. Default 2.
...	Additional arguments passed to kknn::kknn (e.g., ykernel, na.action)

Value

Numeric performance metric

---

wrapper_ranger *Random Forest wrapper for CCI*

Description

Random Forest wrapper for CCI

Usage

```
wrapper_ranger(
  formula,
  data,
  train_indices,
  test_indices,
  metric,
  metricfunc = NULL,
  nthread = 1,
```

```
  mtry = NULL,
  num.trees,
  eps = 1e-15,
  ...
)
```

Arguments

formula	Model formula specifying the dependent and independent variables.
data	Data frame containing the dataset to be used for training and testing the model.
train_indices	A vector of indices specifying the rows in data to be used as the training set.
test_indices	A vector of indices specifying the rows in data to be used as the test set.
metric	Type of metric ("RMSE", "Kappa" or "Log Loss")
metricfunc	Optional user-defined function to calculate a custom performance metric. This function should take the arguments data, model, and test_indices, and return a numeric value representing the performance metric.
nthread	Integer. The number of threads to use for parallel processing. Default is 1.
mtry	Integer. The number of variables to possibly split at in each node. Default is the square root of the number of columns in data.
num.trees	Integer. The number of trees to grow in the random forest.
eps	Small value to avoid log(0) in LogLoss calculations. Default is 1e-15.
...	Additional arguments passed to the ranger function.

Value

A numeric value representing the performance metric of the model on the test set.

---

wrapper_svm *SVM wrapper for CCI*

Description

SVM wrapper for CCI

Usage

```
wrapper_svm(
  formula,
  data,
  train_indices,
  test_indices,
  metric,
  metricfunc = NULL,
  eps = 1e-15,
  ...
)
```

Arguments

formula	Model formula
data	Data frame
train_indices	Indices for training data
test_indices	Indices for testing data
metric	Type of metric ("RMSE", "Kappa" or "Log Loss")
metricfunc	Optional user-defined function to calculate a custom performance metric.
eps	Small value to avoid log(0) in LogLoss calculations. Default is 1e-15.
...	Additional arguments passed to e1071::svm

Value

Performance metric (RMSE for continuous, Kappa for classification)

---

wrapper_xgboost *Extreme Gradient Boosting wrapper for CCI*

Description

Extreme Gradient Boosting wrapper for CCI

Usage

```
wrapper_xgboost(
  formula,
  data,
  train_indices,
  test_indices,
  metric,
  nrounds = 500,
  metricfunc = NULL,
  nthread = 1,
  eps = 1e-15,
  subsample = 1,
  ...
)
```

Arguments

formula	Model formula
data	Data frame
train_indices	Indices for training data
test_indices	Indices for training data

metric	Type of metric ("RMSE", "Kappa" or "Log Loss")
nrounds	Number of boosting rounds
metricfunc	A user specific metric function which have the arguments data, model test_indices and test_matrix and returns a numeric value
nthread	Integer. Number of threads to use for parallel computation during model training in XGBoost. Default is 1.
eps	Small value to avoid log(0) in LogLoss calculations. Default is 1e-15.
subsample	Numeric. The proportion of the data to be used for subsampling. Default is 1 (no subsampling).
...	Additional arguments passed to xgb.train

Value

Performance metric

Index