

Package: CCI (via r-universe)

October 29, 2026

Type Package

Title Computational Test for Conditional Independence

Version 0.4.0

Date 2026-09-28

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](https://doi.org/10.3390/a17080323)>.

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

Suggests testthat, knitr, rmarkdown, dagitty

License GPL (>= 2)

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

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

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

Repository <https://khliland.r-universe.dev>

Date/Publication 2026-09-29 14:34:47 UTC

RemoteUrl <https://github.com/khliland/cci>

RemoteRef HEAD

RemoteSha 2c71af0ecbcc9432c1b9358e8d7d84dd2e7e0487

Contents

add_interaction_terms	2
add_poly_terms	4
build_formula	5
CCI.crt	5
CCI.direction	8
CCI.pretuner	10
CCI.test	13
check_formula	17
clean_formula	17
ExponentialNoise	18
get_pvalues	19
get_tuned_params	20
HardCase	20
is_categorical_Z_any	21
make_strata_from_categorical_Z	21
NonLinearCategorization	22
NonLinNormal	23
NonLinNormalZs_d0	23
NonLinNormalZs_d05	24
NormalData	25
perm.test	26
permute_within_strata	28
plot.CCI	29
PoissonNoise	30
PolyData	31
print.CCI	31
QQplot	32
reduce_Z_pca	34
test.gen	35
unclean_formula	38
UniformNoise_large	38
wrapper_knn	39
wrapper_ranger	40
wrapper_svm	41
wrapper_xgboost	42
Index	44

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

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

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_Z2`).

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	<i>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).</i>
----------------	--

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. Only the numeric variables among them get polynomial terms; categorical variables (factors) are skipped.
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	<i>Build an expanded formula with poly and interaction terms</i>
---------------	--

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 $Y \sim X \mid Z_1 + Z_2$
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.crt	<i>Conditional randomization version of the CCI test (CRT-CCI, experimental)</i>
---------	--

Description

CCI.crt() tests $Y \perp\!\!\!\perp X \mid Z$ in the same way as [CCI.test](#), but builds the null distribution by drawing X from an estimate of its conditional distribution $P(X \mid Z)$ instead of permuting X . This is the conditional randomization test (CRT) of Candès et al. (2018), with the conditional distribution estimated by a quantile random forest, combined with the Monte Carlo cross-validation of CCI.

Usage

```
CCI.crt(
  formula = NULL,
  data,
  crt_trees = 100,
  crt_min_node_size = 5,
  residual_sampling = FALSE,
  ...
)
```

Arguments

<code>formula, data</code>	As in CCI.test , e.g. $Y \sim X \mid Z_1 + Z_2$.
<code>crt_trees</code>	Integer. Number of trees in the forest that estimates $P(X \mid Z)$. Default is 100.
<code>crt_min_node_size</code>	Integer. Minimal node size of that forest; larger values give smoother estimates of the conditional distribution. Default is 5.
<code>residual_sampling</code>	Logical. If TRUE (numeric X only), each X_i^* is drawn as its estimated conditional mean plus a random residual, $X_i^* = \hat{m}(z_i) + r$, instead of from a quantile regression forest. See Details. Default is FALSE.
<code>...</code>	Other arguments passed to CCI.test , e.g. <code>method</code> , <code>nperm</code> , <code>metric</code> , <code>metricfunc</code> , <code>mlfunc</code> , <code>tail</code> , <code>parametric</code> , <code>MC_sample</code> , <code>tune</code> , <code>seed</code> or <code>progress</code> .

Details

Permuting X, as in [CCI.test](#), breaks both the relation between X and Y and the relation between X and Z. Under H_0 only the X-Y relation should be broken. `CCI.crt()` keeps the X-Z relation by drawing each X_i^* from $\hat{P}(X \mid Z = z_i)$, so that $Y \perp\!\!\!\perp X^* \mid Z$ holds by construction (up to the estimation error of $\hat{P}$). Everything else is as in [CCI.test](#): in each of the `nperm` Monte Carlo samples the data are split into a training and a test part, the learner is fitted with X^* on the training part, and its performance on the test part is one value of the null distribution. The test statistic uses the real X.

The conditional distribution is estimated in each Monte Carlo sample with `ranger`:

- a numeric X uses a quantile regression forest (Meinshausen 2006), and each X_i^* is one random draw from the estimated conditional distribution of X at z_i ;
- a categorical X (factor, character or logical) uses a probability forest, and each X_i^* is a class drawn with the estimated class probabilities.

The draws are cross-fitted: X^* for the test rows is drawn from a forest fitted on the training rows, and for the training rows from a forest fitted on the test rows, so no draw comes from a forest that has seen that observation's own X. The forest uses all conditioning variables, including the polynomial and interaction terms that `CCI.test()` adds. Without conditioning variables ($Y \sim X \mid 1$), $P(X \mid Z) = P(X)$ and X is permuted. When all conditioning variables are categorical (factor, character or logical), X is permuted within the groups of Z, which draws exactly from the empirical $P(X \mid Z)$; this is the same stratified permutation as in `CCI.test(robust = TRUE)`, and no forest is fitted.

A quantile forest draw is the X value of a neighbouring observation in Z -space, so it has extra spread when X changes quickly with Z compared with its noise (X almost a function of Z). The real X is then a sharper proxy for Z than X^* , which can inflate the type I error. With `residual_sampling = TRUE`, a numeric X is instead drawn in two steps, separately for each part of the split (location and noise are estimated on the other part):

1. a random forest estimates the conditional mean $m(z) = E[X \mid Z = z]$;
2. residuals $r = X - \hat{m}(Z)$ are computed out-of-fold with 5-fold cross-fitting, so no residual comes from a model that has seen that observation;
3. $X_i^* = \hat{m}(z_i) + r$, with r drawn at random from these residuals.

This assumes additive noise that does not change with Z . It fits 12 instead of 2 sampler forests per Monte Carlo sample.

Compared with `CCI.test`, the null distribution targets the conditional independence hypothesis more directly, at the cost of estimating $P(X \mid Z)$ (two extra forests per Monte Carlo sample). The argument `robust` has no effect, since the draws already depend on Z .

Value

An object of class 'CCI', as from `CCI.test`. Its settings contain `null_method = "crt"`, so `QQplot` repeats the test in the same way.

Experimental

`CCI.crt()` is experimental: its arguments and defaults may change, and it is not (yet) a replacement for `CCI.test`. In simulations with the CIsimdata scenarios of the SoftwareX article (20 scenarios, 100 data sets each), it had about the same power as `CCI.test()`, but not a lower type I error on average.

When it may help over CCI.test

`CCI.crt()` is expected to help when X depends clearly on Z , but X is not almost a function of Z : X has noticeable noise around its relation to Z , and Z has few variables. Then permuting X breaks a relation between X and Z that matters, which can inflate the type I error of `CCI.test()`, while the conditional distribution of X is estimated well. In the simulations the type I error was lower than with `CCI.test()` in such scenarios (e.g. 0.06 vs 0.10 and 0.09 vs 0.13).

It is not expected to help, and can be worse, when X is almost a deterministic function of Z (the draws of X^* are then less precise than the real X ; `residual_sampling = TRUE` may reduce this), when Z has many variables, or when all conditioning variables are categorical (the two functions then run the same test).

References

- Candès, E., Fan, Y., Janson, L. and Lv, J. (2018). Panning for gold: 'model-X' knockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B*, 80(3), 551-577.
- Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7, 983-999.

See Also

[CCI.test](#), [QQplot](#)

Examples

```
set.seed(1)
n <- 300
Z <- rnorm(n)
X <- Z + rnorm(n, sd = 0.5)
Y <- Z^2 + rnorm(n, sd = 0.5)    # Y depends on Z only
dat <- data.frame(X, Y, Z)
res <- CCI.crt(Y ~ X | Z, data = dat, nperm = 20, progress = FALSE)
res
```

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 $Y \parallel X | Z$, the function return the recommended formula either $Y \sim X | Z$ or $X \sim Y | Z$.

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,
  MC_sample = 1,
  mtry = NULL,
  nthread = 1,
  k = 15,
  kernel = "optimal",
  distance = 2,
  poly = TRUE,
  degree = 3,
  interaction = TRUE,
  verbose = FALSE,
  subsample = NULL,
  ...
)
```

Arguments

formula	A formula of the form $Y \sim X \mid Z1 + Z2$ (or $Y \sim X + Z1 + Z2$, or $Y \sim X \mid 1$ without conditioning variables).
data	A data frame containing the variables specified in the formula.
method	Character. The learner: "rf" (random forest, ranger), "xgboost", "svm" (e1071) or "KNN" (kknn). Default is "rf".
folds	Integer. The number of folds for cross-validation. Default is 4.
nrounds	Integer. The number of trees for "rf" and boosting rounds for "xgboost". Default is 600.
max_depth	Integer. The maximum depth of the trees for xgboost. Default is 6.
eta	Numeric. The learning rate for xgboost. Default is 0.3.
gamma	Numeric. The minimum loss reduction required to make a further partition on a leaf node for xgboost. Default is 0.
colsample_bytree	Numeric. The subsample ratio of columns when constructing each tree for xgboost. Default is 1.
min_child_weight	Numeric. The minimum sum of instance weight (hessian) needed in a child for xgboost. Default is 1.
MC_sample	Numeric between 0 and 1. The share of the data to use (a random sample). Default is 1 (all data).
mtry	Integer. The number of variables tried at each split for "rf". Default is NULL (ranger's default).
nthread	Integer. The number of threads for "rf" and "xgboost". Default is 1.
k	Integer. The number of neighbours for "KNN". Default is 15.
kernel	Character. The kernel for "KNN". Default is "optimal".
distance	Numeric. The Minkowski distance parameter for "KNN". Default is 2.
poly	Not used. Kept for backward compatibility; CCI.test adds polynomial terms before choosing the direction.
degree	Not used. Kept for backward compatibility.
interaction	Not used. Kept for backward compatibility.
verbose	Logical. If TRUE, prints the cross-validated RMSE of both directions and the selected formula. Default is FALSE.
subsample	Deprecated, use MC_sample.
...	Additional arguments passed to the learner's wrapper function.

Details

Both directions are evaluated with k-fold cross-validation, using the same learners as [CCI.test\(wrapper_ranger, wrapper_xgboost, wrapper_svm or wrapper_knn\)](#) and the same folds for both directions. The numeric variables are standardized (mean 0, standard deviation 1) before the models are fitted, so the cross-validated RMSEs of the two directions are comparable and the choice does not depend on the units of the variables. The direction with the lowest standardized RMSE, i.e. the variable that is easiest to predict, is chosen.

Value

The selected formula, either $Y \sim X \mid Z$ or $X \sim Y \mid Z$. With `verbose = TRUE`, the cross-validated standardized RMSE of both directions is printed.

Examples

```
set.seed(1)
dat <- data.frame(Z1 = rnorm(200), Z2 = rnorm(200))
dat$X <- dat$Z1 + dat$Z2 + rnorm(200, sd = 0.1) # X is easy to predict from Z
dat$Y <- dat$Z1 + rnorm(200)                  # Y is harder to predict
CCI.direction(Y ~ X | Z1 + Z2, data = dat, method = "xgboost", nrounds = 50)
```

CCI.pretuner

CCI tuner function for CCI test

Description

The `CCI.pretuner` function performs a (random) grid search over hyperparameters for the machine learning models used by `CCI.test`. Each parameter combination is evaluated with the same model wrappers as used in `CCI.test` (`wrapper_ranger`, `wrapper_xgboost` and `wrapper_svm`), and all combinations are evaluated on the same resampling splits, so differences in performance reflect the parameters and not the random splits.

Usage

```
CCI.pretuner(
  formula,
  data,
  method = "rf",
  metric = "RMSE",
  validation_method = "cv",
  folds = 4,
  training_share = 0.7,
  tune_length = NULL,
  random_grid = TRUE,
  samples = 35,
  poly = TRUE,
  degree = 3,
  interaction = TRUE,
  verboseIter = NULL,
  include_explanatory = FALSE,
  verbose = FALSE,
  progress = TRUE,
  parallel = NULL,
  nthread = 1,
  num_trees = 500,
  mtry = NULL,
```

```

nrounds = seq(100, 1000, by = 100),
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 = NULL,
sigma = seq(0.1, 2, by = 0.3),
C = seq(0.1, 2, by = 0.5),
...
)

```

Arguments

formula	Model formula of the form $Y \sim X \mid Z1 + Z2$ (or $Y \sim X + Z1 + Z2$).
data	A data frame containing the variables specified in the formula.
method	Character. The machine learning method to tune: random forest "rf", extreme gradient boosting "xgboost" or Support Vector Machine "svm". Default is "rf".
metric	Character. The performance metric to optimize: "RMSE" (continuous outcome), "Kappa" or "LogLoss" (categorical outcome). Default is "RMSE".
validation_method	Character. The resampling method. One of "cv" (k-fold cross-validation), "LGOCV" (leave-group-out / Monte Carlo cross-validation) or "boot" (bootstrap, evaluated on the out-of-bag observations). Default is "cv".
folds	Integer. Number of folds ("cv") or resamples ("LGOCV", "boot"). Default is 4.
training_share	Numeric. Training share for validation_method = "LGOCV". Default is 0.7.
tune_length	Deprecated and ignored. Use samples to control the number of parameter combinations tried.
random_grid	Logical. If TRUE, samples combinations are drawn at random from the full grid. If FALSE, the full grid is evaluated. Default is TRUE.
samples	Integer. The number of random combinations to draw from the grid when random_grid = TRUE. Default is 35.
poly	Logical. If TRUE, polynomial terms of the conditioning variables are added. Set to FALSE if data already contains them (as when called from CCI.test). Default is TRUE.
degree	Integer. The degree of polynomial terms to include if poly is TRUE. Default is 3.
interaction	Logical. If TRUE, pairwise interaction terms of the conditioning variables are added. Default is TRUE.
verboseIter	Deprecated and ignored.
include_explanatory	Logical. If TRUE, given the condition $Y \parallel X \mid Z$, the explanatory variable X is included when tuning the model for Y. Default is FALSE.

<code>verbose</code>	Logical. If TRUE, prints details about the tuning process. Default is FALSE.
<code>progress</code>	Logical. If TRUE, shows a progress bar. Default is TRUE.
<code>parallel</code>	Deprecated and ignored. Use <code>nthread</code> for multithreaded model fitting.
<code>nthread</code>	Integer. Number of threads used when fitting rf and xgboost models. Default is 1.
<code>num_trees</code>	Integer. Number of trees used when tuning random forest. Default is 500.
<code>mtry</code>	Integer vector. Candidate values for the number of variables sampled at each split in random forest. Default is NULL, which uses $1:\min(p, \max(5, \text{ceiling}(p/2)))$ where p is the number of predictors.
<code>nrounds</code>	Integer vector. Candidate numbers of boosting rounds for xgboost. Default is <code>seq(100, 1000, by = 100)</code> .
<code>eta</code>	Numeric vector. Candidate learning rates for xgboost. Default is <code>seq(0.01, 0.3, by = 0.05)</code> .
<code>max_depth</code>	Integer vector. Candidate maximum tree depths for xgboost. Default is 2:6.
<code>gamma</code>	Numeric vector. Candidate minimum loss reductions for xgboost. Default is <code>c(0, 1, 2, 3)</code> .
<code>colsample_bytree</code>	Numeric vector. Candidate column subsample ratios for xgboost. Default is <code>c(0.8, 0.9, 1)</code> .
<code>min_child_weight</code>	Numeric vector. Candidate minimum child weights for xgboost. Default is <code>c(1, 3)</code> .
<code>subsample</code>	Deprecated and ignored.
<code>sigma</code>	Numeric vector. Candidate RBF kernel widths for svm (passed as <code>gamma</code> to <code>e1071::svm</code>). Default is <code>seq(0.1, 2, by = 0.3)</code> .
<code>C</code>	Numeric vector. Candidate cost (regularization) values for svm (passed as <code>cost</code> to <code>e1071::svm</code>). Default is <code>seq(0.1, 2, by = 0.5)</code> .
<code>...</code>	Additional arguments passed to the model wrapper. A custom grid can be supplied as <code>tuneGrid</code> , a data frame with one column per parameter (<code>mtry</code> for rf; <code>nrounds</code> , <code>eta</code> , <code>max_depth</code> , <code>gamma</code> , <code>colsample_bytree</code> , <code>min_child_weight</code> for xgboost; <code>sigma</code> , <code>C</code> for svm).

Details

By default the model is tuned for predicting Y from the conditioning set Z only (`include_explanatory = FALSE`), i.e. under the null hypothesis.

Value

A list containing:

- `best_param`: A one-row data frame with the best parameters, their mean performance and its standard deviation across resamples, and the method.
- `tuning_result`: A data frame with all tested parameter combinations and their performance, best first.
- `warnings`: A character vector of warnings and errors issued during tuning.

See Also

[CCI.test](#), [get_tuned_params](#), [perm.test](#)

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",
                      progress = FALSE)

result$best_param
```

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,
  MC_sample = "Auto",
  MC_sample_set = NULL,
  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 = NULL,
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,
subsample = NULL,
subsample_set = NULL,
...
)

```

Arguments

formula	Model formula specifying the relationship between dependent and independent variables. (Ex: $Y \sim X \mid Z1 + Z2$ for $Y \parallel X \mid Z1, Z2$). For an unconditional test of $Y \parallel X$, write $Y \sim X \mid 1$ or $Y \sim X + 1$; $Y \sim X$ alone gives an error.
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 and the conditioning set Z contains any categorical variables (factor, character or logical), X is permuted within the groups defined by the categorical variables in Z (stratified permutation). This keeps the relationship between X and the categorical part of Z under the null hypothesis. If FALSE, X is always permuted over all observations. Default is TRUE.
MC_sample	Character. The share of the data used in each Monte Carlo sample: "Auto" uses $(900/n)^{0.75}$ of the data when $n > 900$ and all data otherwise, "Yes" uses the share given in MC_sample_set, and "No" always uses all data. A different random sample is drawn in each Monte Carlo sample. Default is "Auto".
MC_sample_set	Numeric between 0 and 1. The share of the data used in each Monte Carlo sample when MC_sample = "Yes".
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 custom performance metric: a function function(actual, predictions, ...) returning a single number. Set tail to "right" if higher values mean better predictions and "left" if lower values do. actual is numeric for a numeric Y and a factor for a categorical Y. For a numeric Y, predictions is numeric. For a categorical Y, predictions are the predicted classes (a factor) for methods "rf", "svm" and "KNN", and class probabilities for "xgboost": the probability of the second class level for two classes, or an $n \times K$ matrix with the class levels as column names for more classes. The ... arguments are only passed on if the function accepts them. Default is NULL.

<code>mlfunc</code>	Optional the user can pass a custom machine learning wrapper function to use instead of the predefined methods. Default is NULL.
<code>tail</code>	Character. Specifies whether to calculate left-tailed or right-tailed p-values, depending on the performance metric used. Only applicable if using <code>metricfunc</code> or <code>mlfunc</code> . Default is NA.
<code>tune</code>	Logical. If TRUE, hyperparameters for the specified machine learning method are tuned with CCI.pretuner before testing, and the best parameters are used in the test. Available for methods 'rf', 'xgboost' and 'svm'. Default is FALSE.
<code>samples</code>	Integer. Number of hyperparameter combinations used in tuning. Default is 35.
<code>folds</code>	Integer. The number of folds for cross-validation during the tuning process. Default is 5.
<code>tune_length</code>	Deprecated and ignored. Use <code>samples</code> to control the number of parameter combinations tried in tuning.
<code>k</code>	Integer. The number of nearest neighbors to use for KNN method. Default is 15.
<code>center</code>	Logical. If TRUE, the data will be centered before fitting the model
<code>scale</code>	Logical. If TRUE, the data will be scaled before fitting the model. Default is TRUE.
<code>eps</code>	Numeric. A small value to avoid division by zero in some calculations.
<code>positive</code>	Character. The name of the positive class (KNN) in the data, used for classification tasks. Default is NULL.
<code>kernel</code>	Character. The kernel type to use for KNN method. Default is "optimal".
<code>distance</code>	Numeric. Parameter of Minkowski distance for the "KNN" method. Default is 2.
<code>seed</code>	Integer. Set the seed for reproducing results. Default is NA.
<code>random_grid</code>	Logical. If TRUE, a random grid search is performed. If FALSE, a full grid search is performed. Default is TRUE.
<code>nthread</code>	Integer. The number of threads to use for parallel processing. Default is 1.
<code>verbose</code>	Logical. If TRUE, additional information is printed during the execution of the function. Default is FALSE.
<code>progress</code>	Logical. If TRUE, a progress bar is displayed during the permutation process. Default is TRUE.
<code>subsample</code>	Deprecated, use <code>MC_sample</code> .
<code>subsample_set</code>	Deprecated, use <code>MC_sample_set</code> .
<code>...</code>	Additional arguments to pass to the <code>perm.test</code> 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	<i>Check the formula statement</i>
---------------	------------------------------------

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	<i>Clean and Reformat Formula String</i>
---------------	--

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 $Y \sim X + Z1 + Z2$ or $Y \sim X \mid Z1 + Z2$, or $Y \sim X \mid 1$ or $Y \sim X + 1$ for an unconditional test.

Details

For an unconditional test of $Y \perp\!\!\!\perp X$ (no conditioning variables), the formula must be written $Y \sim X \mid 1$ or $Y \sim X + 1$; both are returned as $Y \sim X \mid 1$. A formula without conditioning variables and without the explicit 1, such as $Y \sim X$, gives an error.

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)
clean_formula(y ~ x + 1) # unconditional test
```

ExponentialNoise	<i>Example dataset: ExponentialNoise</i>
------------------	--

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.

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".

Details

Missing values (NA/NaN) in dist, typically from model fits that failed during the permutations, are removed with a warning, and the p-value is computed from the remaining values. If the test statistic is missing, or fewer than two valid null values remain, NA is returned with a warning.

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	<i>Get the best parameters after tuning with CCI.tuner</i>
------------------	--

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), named as the arguments of [CCI.test](#). Returns NULL for unsupported methods. The xgboost row subsample is not returned, since CCI.test has a deprecated argument subsample (the old name of MC_sample).

HardCase	<i>Example dataset: HardCase</i>
----------	----------------------------------

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	<i>Example dataset: NonLinNormalZs_d05</i>
--------------------	--

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,
  MC_sample = 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,
  subsample = NULL,
  ...
)
```

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.
MC_sample	Numeric between 0 and 1. The share of the data used in each Monte Carlo sample (a new random sample each time). Default is 1 (all data).

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 and the conditioning set Z contains any categorical variables (factor, character or logical), X is permuted within the groups defined by the categorical variables in Z (stratified permutation). This keeps the relationship between X and the categorical part of Z under the null hypothesis. If FALSE, X is always permuted over all observations. Default is TRUE.
metricfunc	Optional custom performance metric: a function function(actual, predictions, ...) returning a single number. Set tail to "right" if higher values mean better predictions and "left" if lower values do. actual is numeric for a numeric Y and a factor for a categorical Y. For a numeric Y, predictions is numeric. For a categorical Y, predictions are the predicted classes (a factor) for methods "rf", "svm" and "KNN", and class probabilities for "xgboost": the probability of the second class level for two classes, or an n x K matrix with the class levels as column names for more classes. The ... arguments are only passed on if the function accepts them. 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.
subsample	Deprecated, use MC_sample.
...	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. The element `settings` holds all arguments passed to `test.gen` (including custom `metricfunc/mlfunc` and model parameters), so that `QQplot` can repeat the test with the same settings.

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

<code>x</code>	vector to permute.
<code>strata</code>	factor-like vector defining strata.
<code>seed</code>	optional seed for reproducibility.
<code>na_action</code>	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'
...	ggplot2 layers or themes to add to the plot, e.g. <code>plot(result, ggplot2::labs(title = "My test"))</code> . Other arguments are ignored with a warning. Since the plot is a ggplot2 object, layers can also be added with <code>+</code> .
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

<code>legend.text</code>	Size of legend text
<code>legend.title</code>	Size of legend title
<code>axis.title.x</code>	Size of x-axis title
<code>axis.title.y</code>	Size of y-axis title
<code>base_size</code>	Base font size

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.CCI

Print and summary methods for the CCI class

Description

print() gives a short overview of a CCI test: the hypothesis tested, the learner and metric, the test statistic and the p-value. summary() gives the same with the number of permutations, the tail of the test and the share of data used in each Monte Carlo sample (MC_sample) on separate lines.

When choose_direction = TRUE changed the direction of the test (e.g. from $Y \sim X \mid Z$ to $X \sim Y \mid Z$), the formula that was tested is shown, together with the formula as given.

Usage

```
## S3 method for class 'CCI'
print(x, digits = 4, ...)

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

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

Arguments

<code>x</code>	Object of class 'CCI' (for print) or 'summary.CCI' (for print.summary.CCI)
<code>digits</code>	Number of significant digits for the test statistic and p-value. Default is 4.
<code>...</code>	Additional arguments to print/summary
<code>object</code>	Object of class 'CCI'

Value

The print methods return their argument invisibly, the summary method returns an object of class 'summary.CCI'.

See Also

[perm.test](#), [plot.CCI](#), [QQplot](#)

Examples

```
set.seed(1)
dat <- data.frame(Z = rnorm(100), X = rnorm(100))
dat$Y <- dat$Z + rnorm(100)
res <- CCI.test(Y ~ X | Z, data = dat, nperm = 20, progress = FALSE)
res
summary(res)
```

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

<code>object</code>	Object of class 'CCI'
<code>title.size</code>	Size of the plot title
<code>axis.text.x</code>	Size of x-axis text
<code>axis.text.y</code>	Size of y-axis text
<code>strip.text.x</code>	Size of x-axis strip text
<code>strip.text.y</code>	Size of y-axis strip text
<code>legend.text</code>	Size of legend text
<code>legend.title</code>	Size of legend title
<code>axis.title.x</code>	Size of x-axis title
<code>axis.title.y</code>	Size of y-axis title
<code>progress</code>	Logical indicating whether to show progress during computation
<code>...</code>	Arguments passed to test.gen . By default the test statistic is recomputed <code>nperm</code> times with exactly the settings used to create <code>object</code> (method, metric, custom <code>metricfunc/mlfunc</code> , model parameters, scaling, etc.). Arguments given here override them, e.g. <code>nperm = 50</code> .

Details

The test statistic (without permutation) is recomputed on `nperm` new random train/test splits, and the p-value of each is computed against the stored null distribution. Under the null hypothesis the p-values should be roughly uniform.

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)
```

reduce_Z_pca	<i>Reduce the conditioning variables Z with principal component analysis</i>
--------------	--

Description

Replaces the numeric conditioning variables in a formula $Y \sim X \mid Z_1 + \dots + Z_p$ by their first principal components, and returns the new data and formula for `CCI.test` (or `CCI.crt`). This can make testing faster and more stable when Z has many variables.

Usage

```
reduce_Z_pca(formula, data, ncomp = NULL, var_explained = 0.9, scale = TRUE)
```

Arguments

formula	A formula of the form $Y \sim X \mid Z_1 + Z_2 + \dots$ (or $Y \sim X + Z_1 + Z_2 + \dots$).
data	A data frame with the variables in the formula.
ncomp	Integer. Number of principal components to keep. Default NULL uses var_explained.
var_explained	Numeric between 0 and 1. The share of the variance of the numeric Z variables that the kept components must explain, when ncomp is NULL. Default 0.9.
scale	Logical. If TRUE (default), the numeric Z variables are scaled to unit variance before the PCA.

Details

Only Z is transformed: Y and X are left as they are, and the principal components are computed from the numeric Z variables alone, so the reduction uses no information about Y or X. Categorical Z variables (factor, character or logical) are kept unchanged. By default the numeric Z variables are centered and scaled before the PCA, so that variables measured in large units do not dominate.

The number of components is ncomp if it is given, otherwise the smallest number that explains at least var_explained of the total variance of the (scaled) numeric Z variables. The components are named Z_PC1, Z_PC2, and so on.

A caution. Conditional independence given Z does not imply conditional independence given only the first principal components of Z. If the dropped components (or non-linear functions of Z that the components do not capture) carry information that links X and Y, the test can reject a true hypothesis. Use a high var_explained, and compare with the test on the full Z when possible. With many components, consider `interaction = FALSE` in `CCI.test()`, since the number of pairwise interaction terms grows quickly.

Value

A list with

- data: the data with the principal components Z_PC1, ... added (the original columns are kept);
- formula: the formula with the numeric Z variables replaced by the components;

- ncomp: the number of components kept;
- var_explained: the share of variance explained by the kept components;
- replaced: the names of the numeric Z variables that were replaced;
- pca: the [prcomp](#) object, e.g. for inspecting the loadings.

See Also

[CCI.test](#)

Examples

```
set.seed(1)
n <- 400
Z <- matrix(rnorm(n * 10), n) # 10 correlated conditioning variables
Z <- Z %%% matrix(runif(100, -1, 1), 10)
colnames(Z) <- paste0("Z", 1:10)
dat <- data.frame(Z, X = Z[, 1] + rnorm(n), Y = Z[, 2] + rnorm(n))
red <- reduce_Z_pca(Y ~ X | Z1 + Z2 + Z3 + Z4 + Z5 + Z6 + Z7 + Z8 + Z9 + Z10, dat)
red$formula
red$ncomp
res <- CCI.test(red$formula, data = red$data, nperm = 20, interaction = FALSE, progress = FALSE)
res
```

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,
  MC_sample = 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,
null_method = "permutation",
crt_trees = 100,
crt_min_node_size = 5,
residual_sampling = FALSE,
subsample = NULL,
...
)

```

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.
MC_sample	Numeric between 0 and 1. The share of the data used in each Monte Carlo sample (a new random sample each time). Default is 1 (all data).
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 and the conditioning set Z contains any categorical variables (factor, character or logical), X is permuted within the groups defined by the categorical variables in Z (stratified permutation). If FALSE, X is always permuted over all observations. Default is TRUE.

metricfunc	Optional custom performance metric: a function function(actual, predictions, ...) returning a single number. Set tail to "right" if higher values mean better predictions and "left" if lower values do. actual is numeric for a numeric Y and a factor for a categorical Y. For a numeric Y, predictions is numeric. For a categorical Y, predictions are the predicted classes (a factor) for methods "rf", "svm" and "KNN", and class probabilities for "xgboost": the probability of the second class level for two classes, or an n x K matrix with the class levels as column names for more classes. The ... arguments are only passed on if the function accepts them. Default is NULL.
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.
null_method	Character. How X is replaced when permutation = TRUE: "permutation" (default) permutes X, as in CCI.test ; "crt" draws X from an estimate of $P(X Z)$ with a quantile random forest (conditional randomization), as in CCI.crt .
crt_trees	Integer. Number of trees in the forest that estimates $P(X Z)$ when null_method = "crt". Default is 100.
crt_min_node_size	Integer. Minimal node size in the forest that estimates $P(X Z)$ when null_method = "crt". Default is 5.
residual_sampling	Logical. When null_method = "crt" and X is numeric, draw X as its estimated conditional mean plus an out-of-fold residual instead of from a quantile forest. See CCI.crt . Default is FALSE.
subsample	Deprecated, use MC_sample.
...	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	<i>Convert CI-style formula $Y \sim X \mid Z$ into regression-style $Y \sim X + Z$</i>
-----------------	--

Description

Convert CI-style formula $Y \sim X \mid Z$ into regression-style $Y \sim X + Z$

Usage

```
unclean_formula(f)
```

Arguments

f A formula of the form $Y \sim X \mid Z1 + Z2$, or already $Y \sim X + Z1 + Z2$.

Value

A standard formula of the form $Y \sim X + Z1 + Z2$.

UniformNoise_large	<i>Example dataset: UniformNoise_large</i>
--------------------	--

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	<i>k-Nearest Neighbors (KNN) wrapper for CCI (kkn-based)</i>
-------------	--

Description

k-Nearest Neighbors (KNN) wrapper for CCI (kkn-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), or the name of a custom metric when metricfunc is given |

metricfunc	Optional custom metric function function(actual, predictions, ...). A numeric response gives regression (numeric predictions); a factor, character or logical response gives classification (predicted classes as a factor). The ... arguments are only passed on if the function accepts them.
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 kkn. Default "optimal".
distance	Numeric. Minkowski distance parameter. 2 = Euclidean. Default 2.
...	Additional arguments passed to kkn::kkn (e.g., ykernel, na.action)

Value

Numeric performance metric

wrapper_ranger	<i>Random Forest wrapper for CCI</i>
----------------	--------------------------------------

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 function(actual, predictions, ...) returning a numeric performance value. For a factor response, predictions are the predicted classes. The ... arguments are only passed on if the function accepts them.
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	<i>SVM wrapper for CCI</i>
-------------	----------------------------

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	<i>Extreme Gradient Boosting wrapper for CCI</i>
-----------------	--

Description

Fits an xgboost model on the training rows and returns its performance on the test rows.

Usage

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

Arguments

formula	Model formula
data	Data frame
train_indices	Indices for training data
test_indices	Indices for test data
metric	Type of metric ("RMSE", "Kappa" or "LogLoss"), or the name of a custom metric when metricfunc is given
nrounds	Number of boosting rounds
metricfunc	Optional user-defined function function(actual, predictions, ...) returning a numeric performance value. See Details.
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.
MC_sample	Not used by the model. The share of data per Monte Carlo sample is applied in test.gen before the wrapper is called. (xgboost's own row subsampling can be set with subsample in ...)
...	Additional arguments passed to xgb.train as parameters (e.g. eta, max_depth, objective).

Details

The type of task is decided from the metric and the response: "RMSE" gives regression, "Kappa" and "LogLoss" give classification (binary or multiclass). With a custom `metricfunc`, a numeric response gives regression and a factor, character or logical response gives classification. Classes are encoded as 0..K-1 internally, so any class labels work.

A custom `metricfunc` is called as `metricfunc(actual, predictions, ...)` (with the ... arguments only if the function accepts them), where `actual` has the type of the response (numeric for regression, factor for classification) and `predictions` is a numeric vector for regression, the probability of the second class level for binary classification, and an $n \times K$ probability matrix with the class levels as column names for multiclass classification.

Value

Performance metric

Index

* datasets

- ExponentialNoise, 18
- HardCase, 20
- NonLinearCategorization, 22
- NonLinNormal, 23
- NonLinNormalZs_d0, 23
- NonLinNormalZs_d05, 24
- NormalData, 25
- PoissonNoise, 30
- PolyData, 31
- UniformNoise_large, 38

add_interaction_terms, 2

add_poly_terms, 4

build_formula, 5

CCI (CCI.test), 13

CCI.crt, 5, 34, 37

CCI.direction, 8

CCI.pretuner, 10, 16

CCI.test, 5–10, 13, 13, 20, 34, 35, 37

check_formula, 17

clean_formula, 17

ExponentialNoise, 18

get_pvalues, 19

get_tuned_params, 13, 20

HardCase, 20

is_categorical_Z_any, 21

make_strata_from_categorical_Z, 21

NonLinearCategorization, 22

NonLinNormal, 23

NonLinNormalZs_d0, 23

NonLinNormalZs_d05, 24

NormalData, 25

perm.test, 13, 16, 26, 30, 32, 33

permute_within_strata, 28

plot.CCI, 16, 28, 29, 30, 32, 33

PoissonNoise, 30

PolyData, 31

prcomp, 35

print.CCI, 28, 30, 31, 33

print.summary.CCI, 16

print.summary.CCI (print.CCI), 31

QQplot, 7, 8, 16, 28, 32, 32

reduce_Z_pca, 34

reports (print.CCI), 31

summary.CCI, 28, 30, 33

summary.CCI (print.CCI), 31

test.gen, 28, 33, 35, 42

tuner (CCI.pretuner), 10

unclean_formula, 38

UniformNoise_large, 38

wrapper_knn, 9, 39

wrapper_ranger, 9, 10, 40

wrapper_svm, 9, 10, 41

wrapper_xgboost, 9, 10, 42