Package 'citcdf'

August 21, 2026

Type Package

Title Conditional Independence Testing with Cumulative Distribution Functions

Version 1.1.0

Depends R (>= 4.1)

Imports ggplot2, pbapply, parallel, patchwork, RcppNumerical, survey, viridisLite

Suggests BiocSet, bnlearn, GSA, knitr, quarto, reactable, rmarkdown, sessioninfo, SeuratObject, testthat (>= 3.0.0)

Description
Complex hypothesis testing through conditional cumulative distribution function estimation. Method is detailed in: Gauthier M, Agniel D, Thiébaut R & Hejblum BP (2021). ``Distribution-free complex hypothesis testing for single-cell RNA-seq differential expression analysis", bioRxiv <doi:10.1101/2021.05.21.445165>.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/sistm/citcdf, https://sistm.github.io/citcdf/

BugReports https://github.com/sistm/citcdf/issues

Language en-US

VignetteBuilder quarto

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Denis Agniel [aut],
Kalidou Ba [ctb],
Sara Fallet [aut],
Marine Gauthier [aut],
Boris P. Hejblum [aut, cre, cph],
Pierre Neuvial [ctb]

Maintainer Boris P. Hejblum <boris.hejblum@u-bordeaux.fr>

Repository CRAN

Date/Publication 2026-08-21 13:30:32 UTC

Contents

`ccdf` *Estimate the empirical conditional cumulative distribution function*

Description

Estimate the empirical conditional cumulative distribution function

Usage

```
ccdf(
  Y,
  X,
  Z = NULL,
  method = c("OLS", "logistic"),
  fast = TRUE,
  space_y = FALSE,
  number_y = 10
)
```

Arguments

`Y`	a numeric vector of size `n` containing the preprocessed expressions from `n` samples (or cells).
`X`	a data frame containing numeric or factor vector(s) of size `n` containing the variable(s) to be tested (the condition(s) to be tested).
`Z`	a data frame containing numeric or factor vector(s) of size `n` containing the covariate(s).

method	a character string indicating which method to use to compute the CCDF, either 'OLS' or 'logistic'. Default is 'OLS' for greater computational speed.
fast	a logical flag indicating whether the fast implementation of logistic regression should be used. Only if method == 'logistic'. Default is TRUE.
space_y	a logical flag indicating whether the y thresholds are spaced. When space_y is TRUE, a regular sequence between the minimum and the maximum of the observations is used. Default is FALSE.
number_y	an integer value indicating the number of y thresholds (and therefore the number of regressions) to perform the test. Only used if space_y is TRUE. Default is 10.

Details

The space_y / number_y grid controls both the resolution of the statistic and its computational cost. See cit_multi for details on this trade-off.

Value

A list with the following elements:

- cdf: a vector of the cumulative distribution function of a given gene.
- ccdf: a vector of the conditional cumulative distribution function of a given gene, computed given X. Only if Z is NULL.
- ccdf_nox: a vector of the conditional cumulative distribution function of a given gene, computed given Z only (i.e. X is ignored.). Only if Z is not NULL.
- ccdf_x: a vector of the conditional cumulative distribution function of a given gene, computed given X and Z. Only if Z is not NULL.
- y_sort: a vector of the sorted expression points at which the CDF and the CCDFs are calculated.
- x_sort: a vector of the variables associated with y_sort.
- z_sort: a vector of the covariates associated with y_sort. Only if Z is not NULL.

References

Gauthier M, Agniel D, Thiébaut R & Hejblum BP (2021). Distribution-free complex hypothesis testing for single-cell RNA-seq differential expression analysis, *bioRxiv* 445165. doi:10.1101/2021.05.21.445165.

See Also

plot_compare_ccdf, cit_asymp

Examples

```
set.seed(123)
n <- 500
X <- as.factor(rbinom(n = n, size = 1, prob = 0.5))
Y <- ((X == 1) * rnorm(n = n, 0, 1)) + ((X == 0) * rnorm(n = n, 0.5, 1))
res <- ccdf(Y, data.frame(X = X), method = "OLS")
```

`cit_asymp` *Asymptotic test for conditional independence*

Description

Test the conditional independence of Y and X given Z.

Usage

```
cit_asymp(Y, X, Z = NULL, space_y = FALSE, number_y = 10, design = NULL)
```

Arguments

`Y`	a numeric vector of length `n` to test for conditional independence with `X` adjusted on `Z`
`X`	a data frame of size `n` x `p` of numeric or factor vector(s) containing the variable(s) to be tested for conditional independence against `X` adjusted on `Z`.
`Z`	a data frame of size `n` x `q` of numeric or factor vector(s) containing the covariate(s) to condition the independence test upon.
`space_y`	a logical flag indicating whether the y thresholds are spaced. When `space_y` is `TRUE`, a regular sequence between the minimum and the maximum of the observations is used. Default is `FALSE`.
`number_y`	an integer value indicating the number of y thresholds (and therefore the number of regressions) to perform the test. Only used if `space_y` is `TRUE`. Default is `10`.
`design`	an optional (and technical) list of design quantities, as returned by the internal `.cit_design(X, Z, n)`. This is used by `cit_multi()`, to loop-call over many genes while building the model matrix, and computing its cross-product and its inverse only once. Default is `NULL`, in which case they are computed from `X` and `Z`. Users should not be using this argument

Details

The `space_y` / `number_y` grid controls both the resolution of the statistic and its computational cost. See `cit_multi` for details on this trade-off.

Value

A data frame with the following elements:

- `raw_pval` contains the raw p-values for a given gene.
- `test_statistic` contains the test statistic for a given gene.

References

Gauthier M, Agniel D, Thiébaut R & Hejblum BP (2021). Distribution-free complex hypothesis testing for single-cell RNA-seq differential expression analysis, *bioRxiv* 445165. doi:10.1101/2021.05.21.445165.

See Also

cit_perm, cit_multi, ccdf

Examples

```
set.seed(123)
X <- as.factor(rbinom(n = 100, size = 1, prob = 0.5))
Y <- ((X == 1) * rnorm(n = 100, 0, 1)) + ((X == 0) * rnorm(n = 100, 0.5, 1))
res_asymp <- cit_asymp(Y, data.frame(X = X))


Z <- as.factor(rbinom(n = 100, size = 1, prob = 0.5))
X <- as.numeric(Z) - 1  + rnorm(n = 100, sd = 1)
r <- 500
Y <- replicate(r, as.numeric(Z) - 1)
YY <- (Y == 1) * rnorm(n = 100 * r, 0, 1) + (Y == 0) * rnorm(n = 100 * r, 0.5, 1)
pvals_sim <- sapply(seq_len(r), function(i) {
  cit_asymp(YY[, i], data.frame(X = X), data.frame(Z = Z))$raw_pval
})
hist(pvals_sim) # well calibrated p-values are uniform under the null
quantile(pvals_sim)
```

---

cit_gsa *Conditional independence test for gene set analysis*

Description

Conditional independence test for gene set analysis

Usage

```
cit_gsa(
  M,
  X,
  Z = NULL,
  geneset,
  test = c("asymptotic", "permutation"),
  n_perm = 100,
  n_perm_adaptive = c(n_perm, n_perm, n_perm * 3, n_perm * 5),
  thresholds = c(0.1, 0.05, 0.01),
  parallel = interactive(),
  n_cpus = max(1L, detectCores(logical = FALSE) - 1L, na.rm = TRUE),
  adaptive = FALSE,
  space_y = TRUE,
  number_y = 10
)
```

Arguments

M	a data.frame or a matrix of size n x r containing the different Y variables to test for conditional independence with X adjusted on Z.
X	a data frame of size n x p of numeric or factor vector(s) containing the variable(s) to be tested for conditional independence against X adjusted on Z. Multiple variables (p>1) are supported by the asymptotic test, and also by the permutation when Z is NULL.
Z	a data frame of size n x q of numeric or factor vector(s) containing the covariate(s) to condition the independence test upon. Multiple covariates (q>1) are only supported by the asymptotic test.
geneset	a vector, a list, a gmt file format or a BiocSet object. If the parameter is • a vector : corresponds to the gene name of the gene set, must be the same as those of the columns of the matrix M • a list : each elements of the list are a gene set with the names of the genes, must be the same as those of the columns of the matrix M • a gmt file format : the genes names of each genes set in the file, must be the same as those of the columns of the matrix M • a BiocSet object : the genes names of each genes set in the object, must be the same as those of the columns of the matrix M
test	a character string indicating whether the 'asymptotic' or the 'permutation' test is computed. Default is 'asymptotic'.
n_perm	the number of permutations. Default is 100. Only used if test == 'permutation'.
n_perm_adaptive	a vector of the increasing numbers of adaptive permutations to be performed when adaptive is TRUE if p-values are below thresholds. length(n_perm_adaptive) should be equal to length(thresholds)+1. Default is c(n_perm, n_perm, n_perm*3, n_perm*5).
thresholds	a vector of the decreasing thresholds to compute adaptive permutations when adaptive is TRUE. length(thresholds) should be equal to length(n_perm_adaptive)-1. Default is c(0.1, 0.05, 0.01).
parallel	a logical flag indicating whether parallel computation should be enabled. Default is TRUE if interactive() is TRUE, else is FALSE.
n_cpus	an integer indicating the number of cores to be used for the computations. Default is max(1L, parallel::detectCores(logical = FALSE) - 1L, na.rm = TRUE). If n_cpus = 1, then sequential computations are used without any parallelization.
adaptive	a logical flag indicating whether adaptive permutations should be performed. Default is FALSE (unlike cit_multi()). Only used if test == 'permutation'.
space_y	a logical flag indicating whether the y thresholds are spaced out. When space_y is TRUE, a regular sequence between the minimum and the maximum of the observations is used. If FALSE, each unique observed expression value is used as a distinct threshold. Default is TRUE.
number_y	an integer value indicating the number of y thresholds (and therefore the number of regressions) to perform the test. Only used if space_y is TRUE. Default is 10.

Details

The gene-set statistic is the sum of per-gene statistics. For the permutation test, it is computed with each single permutation of X shared and applied across all genes in a set (so inter-gene correlation is preserved).

For the asymptotic test, the null covariance of the stacked threshold indicators is estimated empirically (`crossprod(temp) / n`). The closed form $min(p_i, p_j) - p_i p_j$ used by `cit_asymp` only holds within a gene (because the product of two threshold indicators of the same `Y` is itself an indicator). Across two genes, that same expectation is their joint distribution function, which the marginal proportions do not determine as there is inter-gene correlation present. The empirical estimator coincides with the closed form on the within-gene diagonal blocks, and additionally supplies the between-gene blocks which carry the inter-gene correlation needed by the summed gene-set statistic.

The `space_y` / `number_y` grid controls both the resolution of the statistic and its computational cost. See `cit_multi` for details on this trade-off.

Value

A list with the following elements:

- `which_test`: a character string carrying forward the value of the '`test`' argument indicating which test was performed (either 'asymptotic' or 'permutation').
- `n_perm`: an integer carrying forward the value of the '`n_perm`' argument or '`n_perm_adaptive`' indicating the number of permutations performed (`NA` if asymptotic test was performed).
- `pvals`: computed p-values. A data frame with one row for each gene set, and with 2 columns: the first one '`raw_pval`' contains the raw p-values, the second one '`adj_pval`' contains the FDR adjusted p-values using Benjamini-Hochberg correction. When '`test == "asymptotic"`', a third column '`test_statistic`' contains the gene set test statistics. Gene sets with no gene observed in `M` yield a warning and `NA` in every column; gene sets only partially observed yield a warning and are tested on the measured genes alone.
- `type`: a character string equal to `"gsa"`, identifying the object as the result of a gene set analysis.

See Also

`cit_multi`, `plot.cit_gsa`

Examples

```
# Two conditions and 30 genes, split into two sets: in "responder" every of
# the 10 genes shifts slightly with X, in "null" none of the remaining 20 does.
set.seed(123)
n <- 100
X <- data.frame(X = as.factor(rbinom(n, size = 1, prob = 0.5)))
M <- matrix(rnorm(n * 30), nrow = n, dimnames = list(NULL, paste0("g", 1:30)))
M[, 1:10] <- M[, 1:10] + 0.3 * (as.numeric(X$X) - 1)
geneset <- list(responder = paste0("g", 1:10), null = paste0("g", 11:30))

res <- cit_gsa(M = M, X = X, geneset = geneset,
```

```
  test = "asymptotic", parallel = FALSE)
res$pvals

# Single gene shifts are too small to be detected on their own,
# but together the set is significant.
per_gene <- cit_multi(M = as.data.frame(M[, 1:10]), X = X,
  test = "asymptotic", parallel = FALSE)
min(per_gene$pvals$adj_pval)  # no single gene survives the correction
res$pvals["responder", ]      # the set does

# a single gene set may be given as a plain character vector of M colnames
cit_gsa(M = M, X = X, geneset = paste0("g", 1:10),
  test = "asymptotic", parallel = FALSE)$pvals

# adjusting for a covariate
Z <- data.frame(Z = rnorm(n))
cit_gsa(M = M, X = X, Z = Z, geneset = geneset,
  test = "asymptotic", parallel = FALSE)$pvals


# The permutation test applies each single permutation of X to every gene of a
# set at once, so the correlation between genes is carried into the null.
cit_gsa(M = M, X = X, geneset = geneset,
  test = "permutation", n_perm = 100, parallel = FALSE)$pvals

# genes listed in a set but absent from M are dropped, with a warning
cit_gsa(M = M, X = X,
  geneset = list(partly_measured = c(paste0("g", 1:5), "absent1")),
  test = "asymptotic", parallel = FALSE)$pvals
```

`cit_multi` *Multiple conditional independence testing*

Description

Multiple conditional independence testing

Usage

```
cit_multi(
  M,
  X,
  Z = NULL,
  test = c("asymptotic", "permutation"),
  n_perm = 100,
  n_perm_adaptive = c(n_perm, n_perm, n_perm * 3, n_perm * 5),
  thresholds = c(0.1, 0.05, 0.01),
  parallel = interactive(),
```

```
  n_cpus = max(1L, detectCores(logical = FALSE) - 1L, na.rm = TRUE),
  adaptive = TRUE,
  space_y = TRUE,
  number_y = 10
)
```

Arguments

M	a data.frame or a matrix of size n x r containing the different Y variables to test for conditional independence with X adjusted on Z
X	a data frame of size n x p of numeric or factor vector(s) containing the variable(s) to be tested for conditional independence against X adjusted on Z. Multiple variables (p>1) are supported by the asymptotic test, and also by the permutation when Z is NULL.
Z	a data frame of size n x q of numeric or factor vector(s) containing the covariate(s) to condition the independence test upon. Multiple covariates (q>1) are only supported by the asymptotic test.
test	a character string indicating whether the 'asymptotic' or the 'permutation' test is computed. Default is 'asymptotic'.
n_perm	the number of permutations. Default is 100. Only used if test == 'permutation'.
n_perm_adaptive	a vector of the increasing numbers of adaptive permutations to be performed when adaptive is TRUE if p-values are below thresholds. length(n_perm_adaptive) should be equal to length(thresholds)+1. Default is c(n_perm, n_perm, n_perm*3, n_perm*5).
thresholds	a vector of the decreasing thresholds to compute adaptive permutations when adaptive is TRUE. length(thresholds) should be equal to length(n_perm_adaptive)-1. Default is c(0.1, 0.05, 0.01).
parallel	a logical flag indicating whether parallel computation should be enabled. Default is TRUE if interactive() is TRUE, else is FALSE.
n_cpus	an integer indicating the number of cores to be used for the computations. Default is max(1L, parallel::detectCores(logical = FALSE) - 1L, na.rm = TRUE). If n_cpus = 1, then sequential computations are used without any parallelization.
adaptive	a logical flag indicating whether adaptive additional permutations should be performed. Default is TRUE. Only used if test == 'permutation'. Note cit_gsa() defaults to adaptive = FALSE whereas cit_multi() defaults to TRUE.
space_y	a logical flag indicating whether the y thresholds are spaced out. When space_y is TRUE, a regular sequence between the minimum and the maximum of the observations is used. If FALSE, each unique observed expression value is used as a distinct threshold. Default is TRUE.
number_y	an integer value indicating the number of y thresholds (and therefore the number of regressions) to perform the test. Only used if space_y is TRUE. Default is 10.

Details

With space_y = FALSE the test statistic uses every distinct observed in Y), but its computation cost represents length(unique(Y)) regressions per gene. With space_y = TRUE, it uses instead a regular grid of number_y points, trading resolution for a computational cost that is independent of n. The grid runs from the smallest non-zero observation (a mass of exact zeros, as in count data, does not consume grid points) to max(Y). Raising number_y brings p-values closer towards their space_y = FALSE values.

Value

A list with the following elements:

- which_test: a character string carrying forward the value of the 'test' argument indicating which test was performed (either 'asymptotic' or 'permutation').
- n_perm: an integer carrying forward the value of the 'n_perm' argument or 'n_perm_adaptive' indicating the number of permutations performed (NA if asymptotic test was performed).
- pvals: computed p-values. A data frame with one row for each gene, and with 2 columns: the first one 'raw_pval' contains the raw p-values, the second one 'adj_pval' contains the FDR adjusted p-values using Benjamini-Hochberg correction. When 'test == "asymptotic"', a third column 'test_statistic' contains the gene-wise test statistics.

References

Gauthier M, Agniel D, Thiébaut R & Hejblum BP (2021). Distribution-free complex hypothesis testing for single-cell RNA-seq differential expression analysis, *bioRxiv* 445165. doi:10.1101/2021.05.21.445165.

See Also

cit_asymp, cit_perm, ccdf

Examples

```
set.seed(123)
Z <- as.factor(rbinom(n = 100, size = 1, prob = 0.5))
X <- as.numeric(Z) - 1  + rnorm(n = 100, sd = 1)
r <- 500
Y <- replicate(r, as.numeric(Z) - 1)
Y <- (Y == 1) * rnorm(n = 100 * r, 0, 1) + (Y == 0) * rnorm(n = 100 * r, 0.5, 1)
res_asymp_unadj <- cit_multi(M = data.frame(Y = Y),
  X = data.frame(X = X),
  test = "asymptotic", parallel = FALSE)
mean(res_asymp_unadj$pvals$raw_pval < 0.05)
hist(res_asymp_unadj$pvals$raw_pval)

res_asymp_adj <- cit_multi(M = data.frame(Y = Y),
  X = data.frame(X = X),
  Z = data.frame(Z = Z),
  test = "asymptotic", parallel = FALSE)
```

```
mean(res_asymp_adj$pvals$raw_pval < 0.05)
hist(res_asymp_adj$pvals$raw_pval)

n <- 100
r <- 500
Z1 <- rbinom(n, size = 1, prob = 0.5)
Z2 <- rnorm(n) # rbinom(n, size=1, prob=0.5) + rnorm(n, sd=0.05)
X1 <- Z2 + rnorm(n, sd = 0.2)
X2 <- rnorm(n)
cor(X1, Z2)
Y <- replicate(r, Z2) + rnorm(n * r, 0, 3)
range(cor(Y, Z2))
range(cor(Y, X2))
res_asymp_unadj <- cit_multi(M = data.frame(Y = Y),
  X = data.frame(X1 = X1, X2 = X2),
  test = "asymptotic", parallel = FALSE)
mean(res_asymp_unadj$pvals$raw_pval < 0.05)
hist(res_asymp_unadj$pvals$raw_pval)

res_asymp_adj <- cit_multi(M = data.frame(Y = Y),
  X = data.frame(X1 = X1, X2 = X2),
  Z = data.frame(Z1 = Z1, Z2 = Z2),
  test = "asymptotic", parallel = FALSE)
mean(res_asymp_adj$pvals$raw_pval < 0.05)
hist(res_asymp_adj$pvals$raw_pval)

# permutation test, on a subset of the genes to keep the example short
res_perm_unadj <- cit_multi(M = data.frame(Y = Y[, 1:20]),
  X = data.frame(X1 = X1),
  test = "permutation", adaptive = FALSE, n_perm = 50,
  parallel = FALSE)
mean(res_perm_unadj$pvals$raw_pval < 0.05)


# adaptive permutations spend extra stages only on the smallest p-values
res_perm_adj <- cit_multi(M = data.frame(Y[, 1:20]), # data.frame(Y),
  X = data.frame(X = X),
  Z = data.frame(Z = Z),
  test = "permutation", n_perm = 50, # 2000,
  parallel = FALSE)
mean(res_perm_adj$pvals$raw_pval < 0.05)
```

`cit_perm` *Permutation test for conditional independence*

Description

Permutation test for conditional independence

Usage

```
cit_perm(
  Y,
  X,
  Z = NULL,
  X_star = NULL,
  n_perm = 100,
  space_y = FALSE,
  number_y = 10
)
```

Arguments

Y	a numeric vector of length n to test for conditional independence with X adjusted on Z
X	a data frame of size n x p of numeric or factor vector(s) containing the variable(s) to be tested for conditional independence against X adjusted on Z. Multivariables X are supported if Z is NULL.
Z	a data.frame of size n x 1 of numeric or factor vector containing the covariate to condition the independence test upon. Multiple covariates are not supported for permutation.
X_star	a list of n_perm permuted designs, as returned by X_perm. Default is NULL, in which case X_perm(X, Z, n_perm = n_perm) is called internally. Supply it explicitly when several outcomes must be scored against the same permutations, or to avoid redrawing them inside a loop; see *Details*.
n_perm	the number of permutations. Default is 100. When X_star is supplied it must hold at least n_perm elements; only the first n_perm are used.
space_y	a logical flag indicating whether the y thresholds are spaced out. When space_y is TRUE, a regular sequence between the minimum and the maximum of the observations is used. Default is FALSE.
number_y	an integer value indicating the number of y thresholds (and therefore the number of regressions) to perform the test. Only used if space_y is TRUE. Default is 10.

Details

The space_y / number_y grid controls both the resolution of the statistic and its computational cost. See cit_multi for details on this trade-off.

Leaving X_star as NULL is the convenient form for a single outcome. Across several outcomes it is not equivalent to supplying one: each call would draw its own permutations, whereas cit_multi and cit_gsa deliberately build one pool with X_perm and reuse it for every gene, so that all genes are scored against the same permuted designs. Pass a shared X_star if you are looping over outcomes yourself.

Value

A data frame with the following elements:

- `score` contains the number of permutations whose test statistic is greater than or equal to the observed one.
- `raw_pval` contains the raw p-values for a given gene computed from `n_perm` permutations.
- `test_statistic` contains the observed test statistic for a given gene. It is the same quantity returned by `cit_asymp`.

References

Gauthier M, Agniel D, Thiébaut R & Hejblum BP (2021). Distribution-free complex hypothesis testing for single-cell RNA-seq differential expression analysis, *bioRxiv* 445165. doi:10.1101/2021.05.21.445165.

See Also

`perm_cont`, `X_perm`, `cit_multi`

Examples

```
set.seed(123)
X <- data.frame(X = as.factor(rbinom(n = 100, size = 1, prob = 0.5)))
Y <- (X$X == 1) * rnorm(100) + (X$X == 0) * rnorm(100, mean = 0.5)

# the permuted designs are drawn internally when X_star is left NULL
res_perm <- cit_perm(Y, X, n_perm = 10)
res_perm

# supplying them explicitly is equivalent, and is what to do when several
# outcomes must be scored against the same permutations
X_star <- X_perm(X, Z = NULL, n_perm = 10)
res_perm_shared <- cit_perm(Y, X, X_star = X_star, n_perm = 10)

# adjusting for a covariate Z
Z <- data.frame(Z = rnorm(100))
res_perm_adj <- cit_perm(Y, X, Z = Z, n_perm = 10)
res_perm_adj
```

`perm_cont` *Permutation procedure when Z is continuous*

Description

Permutation procedure when Z is continuous

Usage

```
perm_cont(X, Z)
```

Arguments

X	a numeric or factor vector of length n containing the variable to be tested (the condition to be tested).
Z	a numeric vector of length n containing the covariate. Multiple variables are not allowed.

Details

X_star is a conditional permutation of X: draws are without replacement, so X_star is a bijective rearrangement of X and its marginal distribution is preserved exactly. The permut for observation i is drawn with a weight that decreases with the distance between fitted values of X given Z, so they are matched on Z.

For binary X the fitted value is a linear-probability approximation of the propensity score (matching on the propensity score balances Z, [Rosenbaum and Rubin 1983] and allows to match on that single number rather than on Z). Of note, this approximation can fall outside [0, 1] when the association between X and Z is strong.

This is the conditional permutation test (CPT) of Berrett et al. (2020), which draws a permutation with probability proportional to the likelihood of the permuted assignment under the conditional law of X given Z. The sequential draw used here approximates that distribution rather than sampling from it exactly. Permuting rather than resampling X, as the conditional randomisation test (CRT) of Candes et al. (2018) does, preserves the empirical marginal distribution of X exactly regardless of the error in the fitted conditional law model (and never produces degenerate single-level designs).

Permutation weights are Gaussian in the distance between fitted values, with bandwidth sd(fit) * n^(-1/3) (polynomial weights leave too much mass on distant candidates, so the neighborhood was not local and the X-Z relationship was not preserved). Of note, the bandwidth exponent is larger than Silverman's 1/5 because his rate is optimized for density estimation but leaves a first-order bias that invalidates inference built on it (Hall, 1992; Armstrong and Kolesar, 2020); o(n^(-1/4)) is the condition Kim et al. (2022) prove for the analogous local permutation test.

Value

X_star a vector of permuted X.

References

Berrett TB, Wang Y, Barber RF, Samworth RJ (2020). The conditional permutation test for independence while controlling for confounders. *Journal of the Royal Statistical Society Series B*, **82**(1), 175-197. doi:10.1111/rssb.12340

Candes E, Fan Y, Janson L, 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. doi:10.1111/rssb.12265

Hemerik J, Goeman JJ (2018). Exact testing with random permutations. *TEST*, **27**(4), 811-825. doi:10.1007/s11749017057 11

Rosenbaum PR, Rubin DB (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, **70**(1), 41-55. doi:10.1093/biomet/70.1.41

Kim I, Neykov M, Balakrishnan S, Wasserman L (2022). Local permutation tests for conditional independence. *The Annals of Statistics*, **50**(6), 3388-3414. doi:10.1214/22AOS2233

Hall P (1992). Effect of bias estimation on coverage accuracy of bootstrap confidence intervals for a probability density. *The Annals of Statistics*, **20**(2), 675-694. doi:10.1214/aos/1176348651

Armstrong TB, Kolesar M (2020). Simple and honest confidence intervals in nonparametric regression. *Quantitative Economics*, **11**(1), 1-39. doi:10.3982/QE1199

See Also

X_perm

Examples

```
set.seed(123)
X <- rbinom(n = 100, size = 1, prob = 0.5)
Z <- rnorm(100, 0, 1)
X_star <- perm_cont(X, Z)
table(X, X_star)
```

---

plot.cit_gsa *Plot the conditional CDFs of every gene in a gene set*

Description

Draws, for one gene set, the conditional CDF of each gene given `X` (and `Z`) as a semi-transparent step function, overlaid with a bold gene-set summary curve per level of `X`. Layout, colors, reference curve and faceting follow `plot_compare_ccdf`, of which this is the many-genes counterpart.

Usage

```
## S3 method for class 'cit_gsa'
plot(
  x,
  M,
  X,
  Z = NULL,
  geneset,
  which_set = 1,
  method = c("OLS", "logistic"),
  fast = TRUE,
  space_y = FALSE,
  number_y = 20,
  discretize = !is.factor(X[, 1]) || (!is.null(Z) && !is.factor(Z[, 1])),
  probs = c(0, 0.25, 0.5, 0.75, 1),
  bin_labels = c("Q1", "Q2", "Q3", "Q4"),
  summary_fun = stats::median,
  alpha = 0.25,
```

```
  linewidth = 0.9,
  n_grid = 200,
  ...
)
```

Arguments

x	an object of class cit_gsa, as returned by cit_gsa.
M	a numeric matrix or data frame of size n x r containing the preprocessed expressions, with gene names as column names. Required: cit_gsa() does not retain the data it was called on.
X	a data frame whose first column is the variable of interest.
Z	a data frame whose first column is the covariate, or NULL.
geneset	the gene set to draw: a character vector of gene names, or a named list of such vectors, in which case which_set picks one.
which_set	index or name selecting the gene set when geneset is a list. Default is the first.
method, fast, space_y, number_y	passed to ccdf.
discretize	a logical flag, following the same logic as in plot_compare_ccdf: continuous variables are quartile-binned and combined into a single interaction factor, so every curve drawn is an exact empirical CDF. Because a pointwise median of monotonic curves is itself monotonic, the summary curve then needs no monotonicity correction. Default is FALSE when X (and Z) are already factors, and TRUE otherwise.
probs, bin_labels	passed to the quartile binning, exactly as in plot_compare_ccdf.
summary_fun	the function used to summarize across genes at each expression value. Default is median.
alpha	opacity of the individual gene curves. Default 0.25.
linewidth	width of the summary and reference curves. Default 0.9.
n_grid	number of points on the shared grid used to summarize across genes. Genes in a set do not share expression values, so all curves are re-evaluated on a common grid first. Default 200.
...	further arguments to be passed.

Value

a ggplot object; when Z is supplied, a patchwork composition stacking the CCDF given X alone (panel A) above the CCDF given X and Z (panel B, faceted by Z).

See Also

plot_compare_ccdf for the single-gene version.

Examples

```
set.seed(123)
n <- 60
X <- data.frame(X = as.factor(rbinom(n, size = 1, prob = 0.5)))

# 20 genes: two sets of 10. Only the genes of set1 depend on X.
M <- matrix(rnorm(n * 20), nrow = n,
  dimnames = list(NULL, paste0("g", 1:20)))
M[, 1:10] <- M[, 1:10] + 1.5 * (as.numeric(X$X) - 1)
geneset <- list(set1 = paste0("g", 1:10), set2 = paste0("g", 11:20))

res <- cit_gsa(M = M, X = X, geneset = geneset,
  test = "asymptotic", parallel = FALSE)

# set1: the two summary curves separate, and the individual genes with them
plot(res, M = M, X = X, geneset = geneset, which_set = "set1")


# set2 is null: the two summary curves stay close to each other and to the
# reference CDF
plot(res, M = M, X = X, geneset = geneset, which_set = "set2")

# with a continuous covariate: quartile-binned and faceted by Z
Z <- data.frame(Z = rnorm(n))
plot(res, M = M, X = X, Z = Z, geneset = geneset, which_set = 1)
plot(res, M = M, X = X, Z = Z, geneset = geneset, which_set = 2)
```

`plot.cit_multi` *Plot of gene-wise p-values*

Description

Plotting the sorted exact p-values along with the Benjamini-Hochberg limit and the nominal threshold

Usage

```
## S3 method for class 'cit_multi'
plot(x, ..., nominal_level = 0.05)
```

Arguments

`x`	an object of class `cit_multi`.
`...`	further arguments to be passed
`nominal_level`	a nominal testing level between 0 and 1. Default is 5%: `0.05`.

Value

a ggplot2 of sorted gene-wise p-values

See Also

cit_gsa, cit_asymp, cit_perm, plot.cit_multi

Examples

```
n <- 100
p <- 50
X1 <- as.factor(rbinom(n = n, size = 1, prob = 0.5))
Y <- replicate(p, ((X1 == 1) * rnorm(n = n, 0, 1)) + ((X1 == 0) * rnorm(n = n, 0.5, 1)))

res_asymp <- cit_multi(M = data.frame(Y = Y),
  X = data.frame(X = X1),
  test = "asymptotic",
  parallel = FALSE)
plot(res_asymp)
```

---

plot_compare_ccdf *Function to plot the CCDF according to the type of X and Z*

Description

Function to plot the CCDF according to the type of X and Z

Usage

```
plot_compare_ccdf(
  Y,
  X,
  Z = NULL,
  method = c("OLS", "logistic"),
  fast = TRUE,
  space_y = FALSE,
  number_y = length(unique(Y[, 1])),
  discretize = !is.factor(X[, 1]) || (!is.null(Z) && !is.factor(Z[, 1])),
  probs = c(0, 0.25, 0.5, 0.75, 1),
  bin_labels = c("Q1", "Q2", "Q3", "Q4")
)
```

Arguments

Y	a data frame whose first column contains the preprocessed expressions from n samples (or cells). Its column name is used as the x-axis label.

X	a data frame whose first column is a numeric or factor vector of size n containing the variable to be tested (the condition to be tested). Its column name is used in the legend keys.
Z	a data frame whose first column is a numeric or factor vector of size n containing the covariate. Multiple variables are not allowed. Its column name is used in the legend keys, the facet strips and the panel A title.
method	a character string indicating which method to use to compute the CCDF, either 'OLS' or 'logistic'. Default is 'OLS' for computational speed.
fast	a logical flag indicating whether the fast implementation of logistic regression should be used. Only if method == 'logistic'. Default is TRUE.
space_y	a logical flag indicating whether the y thresholds are spaced. When space_y is TRUE, a regular sequence between the minimum and the maximum of the observations is used. Default is FALSE.
number_y	an integer value indicating the number of y thresholds (and therefore the number of regressions) used to compute the CCDF. Default is length(unique(Y[, 1])), i.e. one threshold per distinct observed value.
discretize	a logical flag. When TRUE, any continuous variable among X and Z is cut at probs into ordered bins. If Z is not NULL, X and Z are combined into a single interaction factor before calling ccdf. Default is FALSE when X and Z are already factors, and TRUE otherwise.
probs	breakpoints (as quantile probabilities) used to bin a continuous X or Z when discretize = TRUE. Default is quartiles. Ignored for variables that are already factors.
bin_labels	labels for the bins produced by probs. Default is c("Q1", "Q2", "Q3", "Q4"). Ignored for variables that are already factors, and truncated if probs produces fewer bins (ties can collapse quantile breakpoints).

Value

a ggplot object. When Z is supplied and at least one of X and Z is a factor, the returned object is a patchwork composition stacking the CCDF marginal on Z (panel A) above the CCDF given X and Z (panel B).

Discretization of continuous X and Z

ccdf() fits a separate regression at every y threshold. With no constraint linking the the fits across thresholds, fitted $P(Y <= y|X, Z)$ are not necessarily monotonic whenever X and/or Z are continuous variables: at each threshold the observed values for X and Z varies, making the empirical conditioning different. Such ccdf computations across varying covariate values at varying thresholds do not carry any monotonicity guarantee, and become hard to interpret graphically. For this reason, we provide the option to discretize X and Z for graphical representation, in order to ease the interpretation.

When X or Z is continuous it is quartile-binned, and the bin labels show the interval each bin spans using a "less than or equal" sign. That sign is drawn through plotmath rather than as a literal character, so it renders identically on every device (including the classic pdf device). workaround is needed.

A note on when X and Z are both factors

When X and Z are already both factors, ccdf() fits an *additive* model without an interaction term. With more than a handful of levels in either variable this means that the fitted ccdf_x can sometimes display some mild non-monotonicity (in practice this happens only with enough levels on both sides to matter). Set discretize = TRUE to force the saturated interaction encoding.

Examples

```
set.seed(123)
n <- 40
Y  <- data.frame(Y = rnorm(n))
Xf <- data.frame(X = as.factor(rbinom(n, size = 1, prob = 0.5)))
Xc <- data.frame(X = rnorm(n))
Zf <- data.frame(Z = as.factor(rbinom(n, size = 1, prob = 0.5)))
Zc <- data.frame(Z = rnorm(n))

# Z absent, X factor         -- CDF plus one CCDF step per level of X
plot_compare_ccdf(Y, Xf)

# Z absent, X continuous     -- CDF step plus CCDF points
plot_compare_ccdf(Y, Xc)

# Z factor, X factor         -- panel B faceted by Z, steps
plot_compare_ccdf(Y, Xf, Zf)


# Z factor, X continuous     -- panel B faceted by Z, points
plot_compare_ccdf(Y, Xc, Zf)

# Z continuous, X factor     -- panel B not faceted
plot_compare_ccdf(Y, Xf, Zc)

# Z continuous, X continuous -- a single panel, CDF plus both CCDFs
plot_compare_ccdf(Y, Xc, Zc)

# A factor with more than two levels gets one colour per level
X3 <- data.frame(X = as.factor(sample(0:2, n, replace = TRUE)))
plot_compare_ccdf(Y, X3, Zf)

# Forcing the interaction encoding even when X and Z are already factors
plot_compare_ccdf(Y, Xf, Zf, discretize = TRUE)
```

X_perm *Compute the conditional permutations*

Description

Compute the conditional permutations

Usage

```
X_perm(X, Z, n_perm = 100)
```

Arguments

X	a numeric or factor vector of length n containing the variable to be tested (the condition to be tested). Multi-variables X are supported if Z is NULL.
Z	a numeric vector of length n containing the covariate. Multiple variables are not allowed.
n_perm	the number of permutations. Default is 100.

Details

The permutations are conditional on Z:

- When Z is NULL, whole rows of X are permuted uniformly.
- When Z is a factor, integer, logical or character, X is permuted independently within each stratum of Z: the empirical association between X and Z is preserved exactly.
- When Z is continuous, no exact strata exist and the draw is delegated to perm_cont, which matches each observation to a swap of similar fitted value of X given Z.

Value

a list with the permuted label vector for each permutation

See Also

perm_cont, cit_perm

Examples

```
set.seed(123)
X <- rbinom(n = 100, size = 1, prob = 0.5)
Z <- rnorm(100, 0, 1)
X_perm(data.frame(X), data.frame(Z), 100)
```

Index