Package 'HausdorffGoF'

May 15, 2026

Type Package

Version 0.3.0

Title One- And Two-Sample Hausdorff Goodness-of-Fit Test

Description Computes the test statistic and p-values of the one-sample and two-sample Hausdorff (H) goodness-of-fit tests. The H statistic measures the Hausdorff distance under the Chebyshev (l-infinity) metric, between the two cumulative distribution functions (cdfs) underlying the corresponding one-sample and two-sample null hypothesis. It coincides to the side length of the largest axis-aligned square (hypercube) that can be inscribed between the two cdfs. The following cases are covered: (i) one-sample, univariate; (ii) two-sample univariate; and (iii) two-sample bivariate. Exact one-sample p-values are computed in O(n^2 log n) time via the 'Exact-KS-FFT' method of Dimitrova, Kaishev, and Tan (2020) <doi:10.18637/jss.v095.i10>; two-sample p-values are obtained by permutation. A key advantage of the H test is that its sensitivity can be directed towards the left tail, body, or right tail of the distribution by tuning a scale parameter sigma, and therefore maximizing its power which as shown numerically is significantly higher than the power of the classical tests such as the Kolmogorov-Smirnov, Cramer-von Mises, and Anderson-Darling test, especially when the right tail of the distribution is targeted. The sensitivity of the test (left tail, body, or right tail) is governed by two parameters psi1 and psi2, whose values needs to be input. Then the optimal value of the scale parameter sigma is automatically computed.

Depends R (>= 3.5.0)

Imports Rcpp (>= 0.12.12), KSgeneral, stats, utils, withr

LinkingTo Rcpp, RcppEigen

License GPL (>= 3)

URL https://github.com/fakecloudsjy/HausdorffGoF

Encoding UTF-8

NeedsCompilation yes

Author Dimitrina S. Dimitrova [aut],
Yun Jia [aut, cre],
Vladimir K. Kaishev [aut]

Maintainer Yun Jia <yunjia2019@gmail.com>

Repository CRAN

Date/Publication 2026-05-15 20:40:14 UTC

Contents

`HausdorffGoF-package` *One- and Two-Sample Hausdorff Goodness-of-Fit Test*

Description

This package computes the test statistic and p-values of the one-sample and two-sample Hausdorff (H) goodness-of-fit test, proposed by Dimitrova, Jia, and Kaishev (2026a) and Dimitrova, Jia, and Kaishev (2026b) respectively. Exploiting the scale dependence of H, it is shown that the sensitivity of the H test can be controlled, making it left-tail, central body or right-tail sensitive and thereby allowing its power to be optimized for the problem at hand.

Simulation studies further demonstrate that, in terms of statistical power, the H test substantially outperforms classical goodness-of-fit tests such as the Kolmogorov-Smirnov, Cramer-von Mises, and Anderson-Darling tests. The improvement is particularly pronounced when discrepancies in the right tail of the distribution are of primary interest, as is often the case in finance and insurance, economics, the natural sciences, and extreme value theory. These results make the H test a compelling and effective alternative to existing testing procedures.

In the **one-sample** setting, given a random sample $X_1, \ldots, X_n$ of size n with empirical cdf $F_n(x)$ and a prespecified null cdf $F(x)$, the H statistic is defined as the Hausdorff distance between the planar curves (completed graphs) F^c and F_n^c of $F(x)$ and $F_n(x)$ (completed by vertical segments

at jump discontinuities). Based on Lemma 2.1 of Dimitrova, Jia, and Kaishev (2026a), the latter can be expressed as

$$H(F^c, F_n^c) = \sup_{A \in F_n^c} \inf_{B \in F^c} \rho_1(A, B) = \sup_{A \in F^c} \inf_{B \in F_n^c} \rho_1(A, B),$$

where, $\rho_1(\cdot, \cdot)$ is the Chebyshev (ℓ_∞) i.e., $\rho_1(A, B) = \max |x_A - x_B|, |z_A - z_B|$.

In the **two-sample** setting, the null hypothesis is $H_0 : F(x) = G(x)$ for all $x \in R^k$, against the alternative $H_1 : F(x) \neq G(x)$ for at least one x, where $F(x)$ and $G(x)$ are unknown k-dimensional cdfs. The null hypothesis H_0 is tested using two independent random samples $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$ of sizes m and n from $F(x)$ and $G(x)$ respectively.

The two-sample H statistic is then defined as the Hausdorff distance between the completed graphs F_m^c and G_n^c of the empirical cdfs $F_m(x)$ and $G_n(x)$, again under the Chebyshev distance ρ_1. Similarly to the one-sample case, using Lemma 2.4 of Dimitrova, Jia, and Kaishev (2026b), it can be expressed as

$$\mathcal{H}_{m,n} = H(F_m^c, G_n^c) = \sup_{A \in F_m^c} \inf_{B \in G_n^c} \rho_1(A, B) = \sup_{A \in G_n^c} \inf_{B \in F_m^c} \rho_1(A, B).$$

A geometric interpretation given by Theorem 2.5 of Dimitrova, Jia, and Kaishev (2026b) is that the H statistic equals the side length of the largest square (hypercube) that can be inscribed in the region between the two curves (hypersurfaces). In the univariate case this is equivalent to the L\'evy metric between F and F_n or F_m and G_n (see Remark 6 of Dimitrova, Jia, and Kaishev 2026b).

The package **HausdorffGoF** implements efficient algorithms to compute the H statistic in the one-sample (Algorithm 1 of Dimitrova, Jia, and Kaishev 2026a), two-sample univariate (Section 3.1 of Dimitrova, Jia, and Kaishev 2026b) and two-sample bivariate (Appendix A of Dimitrova, Jia, and Kaishev 2026b) cases. Exact one-sample p-values are provided via a rectangle-probability representation (Theorem 4 of Dimitrova, Jia, and Kaishev 2026a), evaluated in $O(n^2 \log(n))$ time using the Exact-KS-FFT method of Dimitrova, Kaishev, and Tan (2020) via `ks_c_cdf_Rcpp` from the package **KSgeneral**. Two-sample p-values are obtained by applying exact or Monte Carlo permutation of the H test (see functions `H_test_2s_1d` and `H_test_2s_2d`).

A key feature of the H test is that its p-values are *not* scale-invariant. This property is used to make the test the left-tail or body or right-tail sensitive and correspondingly optimize the power of the test. This is achieved by rescaling the data with a scale parameter $\sigma > 0$ that is selected as follows.

In the **one-sample** case the scaled statistic is $\mathcal{H}_n(\sigma) = H(F_\sigma^c, F_{n,\sigma}^c)$, where $F_\sigma(x) = F(x/\sigma)$ and $F_{n,\sigma}(x) = F_n(x/\sigma)$. An explicit scale-tuning formula

$$\sigma^* = \frac{\psi_1 - \psi_2}{F^{-1}(\psi_1) - F^{-1}(\psi_2)},$$

depending only on the null distribution F and two probability levels $\psi_1 > \psi_2 \in (0, 1)$, allows the power to be focused on the right tail (e.g. $\psi_1 = 0.99, \psi_2 = 0.95$), the left tail (e.g. $\psi_1 = 0.05, \psi_2 = 0.01$), or the body (e.g. $\psi_1 = 0.70, \psi_2 = 0.40$) of the distribution. Proposition 13 of Dimitrova, Jia, and Kaishev (2026a) shows that $\mathcal{H}_n(\sigma^*)$ and its p-value are then invariant under any further affine rescaling of the data.

In the **two-sample** case no null distribution is assumed, so σ^* cannot be computed from F directly. Instead, the scale is estimated from the pooled sample via the permutation-based rule

$$\sigma^* = \mathrm{E}_{R_{m+n}} \left[\max \left(\frac{\psi_1 - \psi_2}{Q_{X_m^\dagger}(\psi_1) - Q_{X_m^\dagger}(\psi_2)}, \frac{\psi_1 - \psi_2}{Q_{Y_n^\dagger}(\psi_1) - Q_{Y_n^\dagger}(\psi_2)} \right) \,\middle|\, Z_{m+n} \right],$$

where $Q_{X_m^\dagger}(\psi)$ and $Q_{Y_n^\dagger}(\psi)$ are the ψ-quantiles of permuted sub-samples $X_m^\dagger$ and $Y_n^\dagger$ of the pooled sample $Z_{m+n} = \{x_1, \ldots, x_m, y_1, \ldots, y_n\}$ (Equation (49) of Dimitrova, Jia, and Kaishev 2026b). Theorem 4.4 of Dimitrova, Jia, and Kaishev (2026b) shows that σ^* converges in probability to the one-sample rule applied to the pooled distribution $E = \eta F + (1-\eta)G$, and that the critical value of the scale-tuned permutation test is asymptotically equivalent to that of the original H test. Recommended choices of (ψ_1, ψ_2) are the same as in the one-sample case: right tail $(0.99,\ 0.95)$, body $(0.70,\ 0.40)$ or $(0.60,\ 0.30)$, left tail $(0.05,\ 0.01)$.

Scale tuning is integrated directly into `Hausdorff_test` via the `scale_psi` argument, which accepts the pair (ψ_1, ψ_2) and computes σ^* automatically before running the test.

The package provides the following exported functions. For the **one-sample** H test: `H_stat_1s_1d` computes the test statistic, `H_test_1s_1d` computes the exact p-value, and `H_test_c_cdf` is the low-level rectangle-probability engine used by `H_test_1s_1d` and useful when $\hat{q}$ is already available. For the **two-sample** H test: `H_test_2s_1d` performs the univariate permutation test, `H_stat_2s_1d_tr` computes the univariate two-sample statistic via the C++ transformation method, `H_stat_2s_1d_p` computes the univariate two-sample statistic via the C++ projection method, `H_test_2s_2d` performs the bivariate permutation test, and `H_stat_2s_2d` computes the Hausdorff distance between two bivariate empirical cdfs. The **unified interface** `Hausdorff_test` and `Hausdorff_stat` are S3 generics that dispatch to the appropriate underlying function based on the type of input, covering one-sample, two-sample univariate, and two-sample bivariate cases in a single call. Null distributions are specified via `distribution`, which constructs a `"NullDist"` object bundling the cdf, quantile function, and density.

Arguments

No arguments. This is a package-level documentation page.

Details

One-sample H test:

The algorithm to compute the H statistic is based on Lemmas 2 and 3 of Dimitrova, Jia, and Kaishev (2026a). It identifies the locally farthest vertices $\{B_l\}$ of the planar curve F_n^c from the null curve F^c, then for each B_l finds the intersection point E_l of the line $\{(x,z) : x + z = x_{B_l} + z_{B_l}\}$ with F^c, and returns $\max_l \rho_1(B_l, E_l)$. This is implemented in `H_stat_1s_1d`.

The exact p-value $P(H(F^c, F_n^c) > q)$ is computed by `H_test_1s_1d` using the rectangle-probability representation of Theorem 4 of Dimitrova, Jia, and Kaishev (2026a):

$$P(H(F^c, F_n^c) > q) = 1 - P(a_i \le U_{(i)} \le b_i,\ 1 \le i \le n),$$

where $U_{(1)} \le \cdots \le U_{(n)}$ are the order statistics of n i.i.d. Uniform$(0,1)$ random variables, and

$$a_i = F\left(F^{-1}\left(\frac{i}{n} - q\right) - q\right), \quad b_i = F\left(F^{-1}\left(\frac{i-1}{n} + q\right) + q\right).$$

This rectangle probability is evaluated in $O(n^2 \log(n))$ time via `KSgeneral::ks_c_cdf_Rcpp`, with the boundary construction handled by `H_test_c_cdf`. A Monte Carlo alternative (bootstrap p-value) is available by passing `method = "mc"` to `Hausdorff_test`.

Two-sample H test:

The two-sample H statistic $\mathcal{H}_{m,n} = H(F_m^c, G_n^c)$ is computed by `H_stat_2s_1d_tr` using the transformation method of Section 3.1 of Dimitrova, Jia, and Kaishev (2026b). The two staircase curves F_m^c and G_n^c are first modified so that one lies entirely above the other (Lemma 2.13), then the plane is rotated by $\pi/4$ so that $\mathcal{H}_{m,n}$ becomes the supremum of half the vertical difference between the rotated curves (Section 3.1, ibid.), attained over a finite set of transformed vertices. The bivariate version `H_stat_2s_2d` implements the projection method of Appendix A of Dimitrova, Jia, and Kaishev (2026b) via an internal C++ routine.

Because the distribution of $\mathcal{H}_{m,n}$ depends on the unknown underlying distributions F and G, p-values are obtained by the permutation approach of Proposition 3.5 of Dimitrova, Jia, and Kaishev (2026b). The permutation Hausdorff statistic $\mathcal{H}_{m,n}^{\dagger}$ is defined by randomly splitting the pooled sample Z_{m+n} into two groups of sizes m and n, computing H on each split, and averaging over all $C = \frac{(m+n)!}{m!n!}$ splits:

$$P(\mathcal{H}_{m,n}^{\dagger} > q \mid Z_{m+n}) = \frac{1}{C} \sum_{i=1}^{C} \mathbf{1}\left\{ H(F_m^{c\dagger}(\pi_i), G_n^{c\dagger}(\pi_i)) > q \right\}.$$

Theorems 3.2–3.4 of Dimitrova, Jia, and Kaishev (2026b) establish that this permutation p-value is asymptotically equivalent to the true p-value of $\mathcal{H}_{m,n}$ under the null and under fixed or contiguous alternatives, and that the permutation test controls the type I error at the nominal level for any $k \geq 1$. This is implemented in `H_test_2s_1d`, which performs exact enumeration when $C \leq 2.147 \times 10^9$ and Monte Carlo permutation otherwise.

The asymptotic null distribution of the normalised statistic $\sqrt{mn/(m+n)}\,\mathcal{H}_{m,n}^{\dagger}$ is given by Theorem 3.6 of Dimitrova, Jia, and Kaishev (2026b) and involves the supremum of a Brownian bridge weighted by the density of the pooled distribution.

Unified interface:

`Hausdorff_test` and `Hausdorff_stat` are S3 generics that dispatch on the class of their second argument `y`: a `distribution` object triggers the one-sample path; a numeric vector triggers the two-sample univariate path; a matrix or list triggers the two-sample bivariate path. `Hausdorff_test` also accepts `scale_psi` to activate scale tuning in a single call, computing σ^* automatically and returning the augmented `"htest"` object with `$sigma` and `$scale_psi` components.

Value

No return value. This is a package-level documentation page. See the individual function help pages (e.g. `Hausdorff_test`, `Hausdorff_stat`) for the return values of each exported function.

Author(s)

Dimitrina S. Dimitrova <D.Dimitrova@city.ac.uk>, Yun Jia <yunjia2019@gmail.com>, Vladimir K. Kaishev <Vladimir.Kaishev.1@city.ac.uk>

Maintainer: Yun Jia <yunjia2019@gmail.com>

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Vladimir K. Kaishev, Senren Tan (2020). "Computing the Kolmogorov-Smirnov Distribution When the Underlying CDF is Purely Discrete, Mixed or Continuous". *Journal of Statistical Software*, **95**(10): 1–42. doi:10.18637/jss.v095.i10.

`distribution` *Specify a null distribution for the one-sample Hausdorff test*

Description

Creates an object of class `"NullDist"` that bundles the null cumulative distribution function $F(x)$ together with its optional quantile function $F^{-1}(p)$ and density $f(x)$. The resulting object is passed as the second argument `y` to `Hausdorff_test` or `Hausdorff_stat` to trigger the one-sample dispatch path.

Usage

```
distribution(CDF, CDFinverse = NULL, pdf = NULL)
```

Arguments

`CDF`	an R function for the null cdf $F(x)$. This argument is mandatory. The function is passed through `Vectorize` internally, so it does not need to operate on vectors natively.
`CDFinverse`	(optional) an R function for the null quantile function $F^{-1}(p)$. The function is passed through `Vectorize` internally. It is additionally extended with the standard probability-space boundary convention used by `H_test_c_cdf` when evaluating the boundary values a_i and b_i (Theorem 4 of Dimitrova, Jia, and Kaishev 2026a): $$F^{-1}(p) = \begin{cases} -\infty, & \text{if } p < 0, \\ +\infty, & \text{if } p > 1, \\ F^{-1}(p), & \text{otherwise.} \end{cases}$$ Only in-range values of p are passed to the user-supplied function; out-of-range values are assigned the boundary constants directly, so no warnings are produced. When supplied, boundary values in `H_test_c_cdf` are computed directly rather than via root-finding, giving higher numerical precision. Defaults to `NULL`.
`pdf`	(optional) an R function for the null density $f(x)$. The function is passed through `Vectorize` internally. When supplied, Newton–Raphson iterations are used in `H_stat_1s_1d` to solve the equation $x + F(x) = \lambda$, which is faster and more accurate than `uniroot` for smooth densities. Also used to compute σ^* numerically when `CDFinverse` is absent and `scale_psi` is supplied to `Hausdorff_test`. Defaults to `NULL`.

Value

An object of class `"NullDist"`: a named list with elements `CDF`, `CDFinverse`, and `pdf`, all stored in their vectorised and (for `CDFinverse`) boundary-extended forms. A `print` method is provided.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

See Also

`Hausdorff_test`, `Hausdorff_stat`.

Examples

```
## Standard normal null (all three functions supplied)
null_norm <- distribution(CDF        = pnorm,
                          CDFinverse = qnorm,
                          pdf        = dnorm)
null_norm

## Exp(2) null -- only CDF supplied (root-finding used internally)
null_exp2 <- distribution(CDF = function(x) pexp(x, rate = 2))
null_exp2
```

`Hausdorff_stat` *Unified one-sample and two-sample Hausdorff test statistic*

Description

An S3 generic that computes the Hausdorff (H) test statistic without a p-value, dispatching on the class of `y` in the same way as `Hausdorff_test`:

`y` **numeric** Two-sample statistic $\mathcal{H}_{m,n} = H(F_m^c, G_n^c)$ via `H_stat_2s_1d_tr`.

`y` `NullDist` One-sample statistic $H(F^c, F_n^c)$ via `H_stat_1s_1d`.

`y` **function** One-sample shorthand; the bare cdf is passed directly to `H_stat_1s_1d` with `pdf = NULL`.

`y` **matrix or list** Two-sample bivariate statistic $H(F_m^c, G_n^c)$ via `H_stat_2s_2d`. Both `x` and `y` must be two-column matrices (or lists of two numeric vectors) of sizes m and n. When `invariant = TRUE` the order-invariant statistic $H^{\circ}_{m,n} = \max_{1\le i\le 4} H(F^c_{m,i}, G^c_{n,i})$ is returned by maximising over the $2^2 = 4$ sign-flip orderings of the two coordinates (Equation (17) of Dimitrova, Jia, and Kaishev 2026b). List input is converted to a two-column matrix and the same path is followed.

Usage

```
Hausdorff_stat(x, y, tol = 1e-10, ...)

## S3 method for class 'numeric'
Hausdorff_stat(x, y, tol = 1e-10, ...)

## S3 method for class 'NullDist'
Hausdorff_stat(x, y, tol = 1e-10, ...)

## S3 method for class 'function'
Hausdorff_stat(x, y, tol = 1e-10, ...)

## S3 method for class 'matrix'
Hausdorff_stat(x, y, tol = 1e-6, invariant = FALSE, ...)

## S3 method for class 'list'
Hausdorff_stat(x, y, tol = 1e-6, invariant = FALSE, ...)
```

Arguments

x	a numeric vector (univariate case) or a two-column numeric matrix or list of two numeric vectors (bivariate case).
y	one of: (i) a numeric vector (two-sample univariate), (ii) a distribution object (one-sample, full control), (iii) a bare vectorised R function for the null cdf (one-sample shorthand), or (iv) a two-column numeric matrix or list of two numeric vectors (two-sample bivariate).
tol	a numeric value giving the tolerance for the root-finding step (one-sample path, default 1e-10) or for locating omnidirectional jump vertices (bivariate path, default 1e-6).
invariant	logical; bivariate path only. When TRUE the order-invariant statistic $H^{\circ}_{m,n}$ is returned by maximising over the four sign-flip orderings of the two coordinates. Defaults to FALSE.
...	further arguments (currently unused).

Value

A single numeric value: the observed Hausdorff statistic $\hat{q}$.

See Also

distribution, Hausdorff_test, H_stat_1s_1d, H_stat_2s_1d_tr, H_stat_2s_2d.

Examples

```
## --- Univariate two-sample statistic ---------------------------------------
set.seed(1)
Hausdorff_stat(rnorm(40), rnorm(40))
```

```
## --- One-sample statistic: full distribution object -------------------------
set.seed(2)
x <- rexp(50, rate = 1)
null_exp <- distribution(CDF = function(t) pexp(t, rate = 1),
                         pdf = function(t) dexp(t, rate = 1))
Hausdorff_stat(x, null_exp)

## --- One-sample statistic: bare CDF shorthand -------------------------------
set.seed(3)
Hausdorff_stat(rnorm(60), pnorm)

## --- Bivariate two-sample statistic (standard ordering) ---------------------
set.seed(4)
x <- matrix(rnorm(100), ncol = 2)
y <- matrix(rnorm(100), ncol = 2)
Hausdorff_stat(x, y)

## --- Bivariate two-sample statistic (order-invariant) -----------------------
Hausdorff_stat(x, y, invariant = TRUE)

## --- Bivariate two-sample statistic: list input -----------------------------
set.seed(5)
x3 <- list(rnorm(30), rnorm(30))
y3 <- list(rnorm(30), rnorm(30))
Hausdorff_stat(x3, y3)
```

`Hausdorff_test` *Unified one-sample and two-sample Hausdorff goodness-of-fit test*

Description

An S3 generic that performs the Hausdorff (H) goodness-of-fit test and returns an object of class `"htest"`, dispatching on the class of `y`.

Dispatch paths.

`y` `NullDist` One-sample test. `H_test_1s_1d` is called when `method` is `"default"` or `"exact"`, giving the exact rectangle-probability p-value. When `method = "mc"`, a Monte Carlo bootstrap p-value is computed instead: `nboots` samples of size n are drawn from F via the quantile transform and the proportion of bootstrap H statistics exceeding the observed value is returned.

`y` **function** One-sample shorthand. The bare CDF is promoted to a `NullDist` with `CDFinverse = NULL` and `pdf = NULL`, then dispatched as above. Activating `scale_psi` with a bare function is not supported because σ^* requires a quantile function; an informative error directs the user to `distribution`.

`y` **numeric** Two-sample univariate test via `H_test_2s_1d`. `"default"` and `"mc"` use Monte Carlo permutation; `"exact"` enumerates all $\frac{(m+n)!}{m!\,n!}$ permutations.

y **matrix or list** Two-sample bivariate test via H_test_2s_2d. Monte Carlo permutation is used by default; method = "exact" enumerates all $\frac{(m+n)!}{m!\,n!}$ splits (automatically falling back to Monte Carlo if the count exceeds 2.147×10^9). List input is converted to a two-column matrix before proceeding.

Resolution of method **by path.**

"default" One-sample: exact rectangle-probability p-value. Two-sample (univariate and bivariate): Monte Carlo permutation.

"exact" One-sample: exact rectangle-probability p-value (same as "default"). Two-sample univariate: full enumeration of all permutations. Two-sample bivariate: full enumeration of all permutations (falls back to Monte Carlo automatically if sample sizes are too large).

"mc" One-sample: Monte Carlo bootstrap p-value. Bootstrap samples are drawn from F via CDFinverse if supplied in the NullDist object, otherwise by uniroot inversion of CDF. Two-sample (univariate and bivariate): Monte Carlo permutation.

Scale tuning (scale_psi)**.** When scale_psi is supplied, σ^* is computed before the test, the data are rescaled, and the test is run on the scaled inputs.

One-sample (y is NullDist): σ^* uses the closed-form formula of Proposition 13 of Dimitrova, Jia, and Kaishev (2026a),

$$\sigma^* = \frac{\psi_1 - \psi_2}{F^{-1}(\psi_1) - F^{-1}(\psi_2)}.$$

The quantile F^{-1} is evaluated by priority: CDFinverse (if supplied in the NullDist object) $>$ Newton–Raphson (if pdf is supplied) $>$ uniroot applied to CDF. The Newton–Raphson iteration is bounded by max.init steps. The sample is replaced by $\sigma^* x$ and the null distribution by $F_{\sigma^*}(t) = F(t/\sigma^*)$. Only functions that exist in the original NullDist object are scaled; absent ones remain NULL.

Two-sample univariate (y is numeric): σ^* is estimated from the pooled sample by averaging over scale_nperms random splits (Equation (49) of Dimitrova, Jia, and Kaishev 2026b),

$$\sigma^* \approx \frac{1}{B}\sum_{b=1}^{B} \max\left(\frac{\psi_1 - \psi_2}{Q^{(b)}_{X_m^\dagger}(\psi_1) - Q^{(b)}_{X_m^\dagger}(\psi_2)},\ \frac{\psi_1 - \psi_2}{Q^{(b)}_{Y_n^\dagger}(\psi_1) - Q^{(b)}_{Y_n^\dagger}(\psi_2)} \right).$$

Both samples are scaled by the scalar σ^*.

Two-sample bivariate (y is matrix or list): the univariate formula is applied column-wise, yielding a length-2 vector (σ_1^*, σ_2^*). Column j of each matrix is scaled by σ_j^*.

In all cases, scale_psi and σ^* are attached to the returned "htest" object as $scale_psi and $sigma, and the string "(scale-tuned)" is appended to $method.

Usage

```
Hausdorff_test(x, y, method = "default", nboots = 2000, tol = 1e-10,
               scale_psi = NULL, scale_nperms = 1000, max.init = 1000, ...)

## S3 method for class 'NullDist'
Hausdorff_test(x, y, method = "default", nboots = 2000,
                                    tol = 1e-10, scale_psi = NULL,
```

```
                                    scale_nperms = 1000, max.init = 1000, ...)

## S3 method for class 'function'
Hausdorff_test(x, y, method = "default", nboots = 2000,
                                     tol = 1e-10, scale_psi = NULL,
                                     scale_nperms = 1000, max.init = 1000, ...)

## S3 method for class 'numeric'
Hausdorff_test(x, y, method = "default", nboots = 2000,
                                    tol = 1e-10, scale_psi = NULL,
                                    scale_nperms = 1000, ...)

## S3 method for class 'matrix'
Hausdorff_test(x, y, method = "default", nboots = 2000,
                                   tol = 1e-6, scale_psi = NULL,
                                   scale_nperms = 1000, max.init = 1000,
                                   invariant = FALSE, ...)

## S3 method for class 'list'
Hausdorff_test(x, y, method = "default", nboots = 2000,
                                  tol = 1e-6, scale_psi = NULL,
                                  scale_nperms = 1000, max.init = 1000,
                                  invariant = FALSE, ...)
```

Arguments

x	a numeric vector (one-sample or two-sample univariate), or a two-column numeric matrix or list of two numeric vectors (bivariate).
y	one of: (i) a distribution object, (ii) a bare CDF function, (iii) a numeric vector, or (iv) a two-column numeric matrix or list. See Dispatch paths above.
method	"default", "exact", or "mc". See the resolution table in the Description. Defaults to "default".
nboots	a positive integer: number of Monte Carlo replications for the test p-value (one-sample bootstrap or two-sample permutation). Defaults to 2000.
tol	numeric tolerance for root-finding in the one-sample statistic and p-value computation (default 1e-10), or for locating omnidirectional jump vertices in the bivariate statistic (default 1e-6). Also used when scale_psi is supplied and Newton–Raphson inversion is needed to compute σ^*.
scale_psi	NULL (no scaling, default) or a length-2 numeric vector c(psi1, psi2) with $0 < \psi_2 < \psi_1 < 1$. The two values may be supplied in any order; they are sorted internally. Recommended choices (Dimitrova, Jia, and Kaishev 2026a, 2026b): **Right tail** c(0.99, 0.95). **Body** c(0.70, 0.40) or c(0.60, 0.30). **Left tail** c(0.05, 0.01).
scale_nperms	a positive integer: number of Monte Carlo splits used to estimate σ^* in the two-sample paths. Ignored when scale_psi = NULL or in the one-sample path. Defaults to 1000.

`max.init` a positive integer: maximum number of Newton–Raphson iterations when inverting F numerically to compute σ^* in the one-sample path. Ignored when `CDFinverse` is supplied in the `NullDist` object, or when `scale_psi = NULL`. Defaults to `1000`.

`invariant` logical; bivariate path only. When `TRUE`, the order-invariant statistic $H^{\circ}_{m,n}$ is used throughout (observed value and all permutation replicates), maximising over the $2^2 = 4$ sign-flip orderings of the two coordinates (Equation (17) of Dimitrova, Jia, and Kaishev 2026b). Defaults to `FALSE`.

`...` further arguments (currently unused).

Value

An object of class `"htest"` with components:

`statistic` the observed Hausdorff statistic $\hat{q}$ (computed on the scaled data when `scale_psi` is supplied), named `"H"`.

`p.value` the p-value computed according to the active path and `method`.

`method` a character string identifying the procedure. The suffix `"(scale-tuned)"` is appended when `scale_psi` is supplied.

`alternative` the character string `"two-sided"`.

`data.name` a character string giving the names of the data objects.

`sigma` (only when `scale_psi` is supplied) the computed σ^*: a scalar for the one-sample and two-sample univariate paths; a length-2 vector for the bivariate path.

`scale_psi` (only when `scale_psi` is supplied) the validated sorted `c(psi_low, psi_high)` vector used to compute σ^*.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Vladimir K. Kaishev, Senren Tan (2020). "Computing the Kolmogorov-Smirnov Distribution When the Underlying CDF is Purely Discrete, Mixed or Continuous". *Journal of Statistical Software*, **95**(10): 1–42. doi:10.18637/jss.v095.i10.

See Also

`distribution`, `Hausdorff_stat`, `H_test_1s_1d`, `H_test_c_cdf`, `H_test_2s_1d`, `H_stat_2s_2d`.

Examples

```
## ---- One-sample, no scaling ------------------------------------------------
set.seed(1)
x      <- rexp(50, rate = 1)
null_e <- distribution(CDF        = function(t) pexp(t, rate = 1),
                       CDFinverse = function(p) qexp(p, rate = 1),
```

```
                        pdf        = function(t) dexp(t, rate = 1))
Hausdorff_test(x, null_e)

## ---- One-sample, scale-tuned: right tail -----------------------------------
res <- Hausdorff_test(x, null_e, scale_psi = c(0.99, 0.95))
res$method    # "... (scale-tuned)"
res$sigma
res$scale_psi



## ---- One-sample, Monte Carlo p-value, scale-tuned: body --------------------
Hausdorff_test(x, null_e, method = "mc", nboots = 1000,
               scale_psi = c(0.70, 0.40))


## ---- One-sample, only CDF and pdf supplied (Newton-Raphson sigma*) ---------
null_cdf_only <- distribution(CDF = function(t) pexp(t, rate = 1),
                              pdf = function(t) dexp(t, rate = 1))
Hausdorff_test(x, null_cdf_only, scale_psi = c(0.99, 0.95))

## ---- Two-sample univariate, no scaling -------------------------------------
set.seed(2)
x1 <- rnorm(40);  x2 <- rnorm(40)
Hausdorff_test(x1, x2)


## ---- Two-sample univariate, scale-tuned ------------------------------------
res2 <- Hausdorff_test(x1, x2, scale_psi = c(0.99, 0.95), scale_nperms = 500)
res2$sigma

## ---- Two-sample univariate, exact permutation, small samples ---------------
set.seed(3)
Hausdorff_test(rnorm(8), rnorm(8), method = "exact")


## ---- Two-sample bivariate, no scaling --------------------------------------
set.seed(4)
xm <- matrix(rnorm(100), ncol = 2)
ym <- matrix(rnorm(100, mean = 0.5), ncol = 2)
Hausdorff_test(xm, ym, nboots = 1000)

## ---- Two-sample bivariate, scale-tuned (column-wise sigma*) ----------------
res3 <- Hausdorff_test(xm, ym, nboots = 1000,
                       scale_psi = c(0.70, 0.40), scale_nperms = 500)
res3$sigma   # length-2 vector, one sigma* per coordinate

## ---- Two-sample bivariate, order-invariant statistic -----------------------
Hausdorff_test(xm, ym, nboots = 1000, invariant = TRUE)
```

`H_stat_1s_1d` *Computes the one-sample Hausdorff test statistic for a given sample and a prespecified null distribution*

Description

Computes the Hausdorff distance $H(F^c, F_n^c)$ between the planar curve F^c of a prespecified null cdf $F(x)$ and the planar curve F_n^c of the empirical cdf based on a given sample $\{x_1, \ldots, x_n\}$.

Usage

```
H_stat_1s_1d(x, CDF, pdf = NULL, tol = 1e-10, max.init = 1000)
```

Arguments

`x`	a numeric vector of data sample values $\{x_1, \ldots, x_n\}$.
`CDF`	a vectorised R function for the null cdf $F(x)$.
`pdf`	(optional) a vectorised R function for the null density $f(x)$. When supplied, Newton–Raphson iterations are used to solve the equation $x + F(x) = \lambda$ in Algorithm 1, which is faster and more accurate than bisection via `uniroot` for smooth densities. Defaults to `NULL`.
`tol`	a numeric value giving the tolerance for the root-finding step. Defaults to `1e-10`.
`max.init`	maximum number of Newton–Raphson iterations. Used only when a density `pdf` is supplied. Defaults to `1000`.

Details

Given a random sample $\{x_1, \ldots, x_n\}$ with empirical cdf $F_n(x)$ and a prespecified null cdf $F(x)$, `H_stat_1s_1d` computes the Hausdorff test statistic

$$H(F^c, F_n^c) = \max_l \rho_1(B_l, E_l),$$

where $\{B_l\}$ are the locally farthest vertices of the planar curve F_n^c from the null curve F^c, E_l is the intersection of the line $\{(x, z) : x + z = x_{B_l} + z_{B_l}\}$ with F^c, and $\rho_1(A, B) = \max\{|x_A - x_B|, |z_A - z_B|\}$. This implements Algorithm 1 (Lemmas 2 and 3) of Dimitrova, Jia, and Kaishev (2026a).

The function first identifies the set $\mathcal{B}_{loc}$ of locally farthest vertices, then for each $B_l \in \mathcal{B}_{loc}$ solves the equation $x + F(x) = x_{B_l} + z_{B_l}$ to find E_l, and finally returns the maximum of $|x_l^* - x_{B_l}|$. When `pdf` is supplied the equation is solved via Newton–Raphson; otherwise `uniroot` is used.

Value

A single numeric value: the observed Hausdorff statistic $\hat{q} = H(F^c, F_n^c)$.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

See Also

H_test_1s_1d, Hausdorff_stat.

Examples

```
## Compute the H statistic for a sample from Exp(1) against the Exp(1) null
set.seed(1)
x <- rexp(50, rate = 1)
H_stat <- H_stat_1s_1d(x   = x,
                       CDF = function(t) pexp(t, rate = 1),
                       pdf = function(t) dexp(t, rate = 1))
H_stat

## Compute the H statistic for a sample from N(0,1) against the N(0,1) null
set.seed(2)
y <- rnorm(100)
H_stat2 <- H_stat_1s_1d(x   = y,
                        CDF = pnorm,
                        pdf = dnorm)
H_stat2
```

H_stat_2s_1d_p *C++ implementation of the two-sample Hausdorff distance via the projection method*

Description

A C++ implementation (via **Rcpp**) of the projection method for computing the Hausdorff distance $H(F_m^c, G_n^c)$ between the planar curves of the empirical cdfs of two independent univariate samples $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$. See Appendix E of Dimitrova, Jia, and Kaishev (2026b) for a numerical accuracy and timing comparison with H_stat_2s_1d_tr.

Usage

```
H_stat_2s_1d_p(a, b)
```

Arguments

a	a numeric vector of data sample values $\{x_1, \ldots, x_m\}$.
b	a numeric vector of data sample values $\{y_1, \ldots, y_n\}$.

Details

H_stat_2s_1d_p implements the **projection method** of Section 3.1 of Dimitrova, Jia, and Kaishev (2026b) and proceeds in five steps identical to those described for the univariate case in the bivariate function H_stat_2s_2d, specialised to $k = 1$. The overall computation is $O(m + n)$ in the number of distinct observations.

Value

A single numeric value: the Hausdorff distance $H(F_m^c, G_n^c)$.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted*.

See Also

`H_stat_2s_1d_tr` for the faster transformation-method implementation.

Examples

```
set.seed(1)
a <- rnorm(50)
b <- rnorm(50)
H_stat_2s_1d_p(a, b)

## Verify agreement with the transformation-method C++ implementation
H_stat_2s_1d_tr(a, b)
```

`H_stat_2s_1d_tr` *C++ implementation of the two-sample Hausdorff distance via the transformation method*

Description

A C++ implementation (via **Rcpp**) of the transformation method for computing the Hausdorff distance $H(F_m^c, G_n^c)$ between the planar curves of the empirical cdfs of two independent univariate samples $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$. This is the faster of the two exported C++ routines; see Appendix E of Dimitrova, Jia, and Kaishev (2026b) for timing comparisons.

Usage

```
H_stat_2s_1d_tr(a, b)
```

Arguments

`a`	a numeric vector of data sample values $\{x_1, \ldots, x_m\}$.
`b`	a numeric vector of data sample values $\{y_1, \ldots, y_n\}$.

Details

H_stat_2s_1d_tr implements the **transformation method** of Section 3.1 of Dimitrova, Jia, and Kaishev (2026b). The plane is rotated by $\pi/4$ via the linear map $\mathrm{Tr}(x, z) = (x - z,\ x + z)$ (Equation (35), ibid.), after which the Hausdorff distance equals half the supremum of the absolute vertical difference between the two rotated staircase curves (Equation (36), ibid.). The algorithm proceeds in four steps.

Step 1 (signed-increment encoding). For each sample, the sorted distinct values and their relative jump heights are extracted. Both samples are shifted by subtracting the joint minimum. Each staircase curve is encoded as an interleaved signed-increment vector.

Step 2 (rotated coordinates via cumulative sums). The $\pi/4$ rotation is performed implicitly: the z'-coordinate $(x + z)$ and the $-x'$-coordinate $(z - x)$ of every staircase vertex are derived from the encoded vectors via two cumulative sums.

Step 3 (greedy pointer sweep). A single left-to-right pass through both encoded curves simultaneously identifies, for each vertex of one curve, the segment of the other curve whose λ-projection interval contains it.

Step 4 (distance formula and return value). At each vertex, the candidate Hausdorff distance is computed in closed form from the parity of the current pointer position. The Hausdorff distance is

$$H(F_m^c, G_n^c) = \frac{1}{2} \max_i |H_{\mathrm{temp}}[i]|.$$

The entire computation is $O(m + n)$ in the number of distinct observations.

Value

A single numeric value: the Hausdorff distance $H(F_m^c, G_n^c)$.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted*.

See Also

H_stat_2s_1d_p for the C++ projection-method implementation (same result, slower).

Examples

```
set.seed(1)
a <- rnorm(50)
b <- rnorm(50)
H_stat_2s_1d_tr(a, b)

## Verify agreement with the projection-method C++ implementation
H_stat_2s_1d_p(a, b)
```

`H_stat_2s_2d` *Computes the Hausdorff distance between two bivariate empirical cdfs*

Description

Computes the Hausdorff distance between the bivariate empirical cdfs of two independent two-dimensional data samples $\boldsymbol{X} = \{(X_{1,1}, X_{1,2}), \ldots, (X_{m,1}, X_{m,2})\}$ and $\boldsymbol{Y} = \{(Y_{1,1}, Y_{1,2}), \ldots, (Y_{n,1}, Y_{n,2})\}$.

Usage

```
H_stat_2s_2d(x, y, tol = 1e-6)
```

Arguments

`x`	a two-column numeric matrix representing the first bivariate sample of size m. For list input use `Hausdorff_stat`.
`y`	a two-column numeric matrix representing the second bivariate sample of size n. For list input use `Hausdorff_stat`.
`tol`	a numeric value giving the tolerance for locating omnidirectional jump vertices in the bivariate empirical cdfs (see Definition 2.11 of Dimitrova, Jia, and Kaishev 2026b). Defaults to `1e-6`.

Details

`H_stat_2s_2d` computes the Hausdorff distance $H(F_m^c, G_n^c)$ between the three-dimensional staircase surfaces corresponding to the bivariate empirical cdfs $F_m(\boldsymbol{x})$ and $G_n(\boldsymbol{x})$ of `x` and `y`, under the Chebyshev distance $\rho_1(A, B) = \max_{1 \le i \le 3}\{|w_A^{(i)} - w_B^{(i)}|\}$. It implements the projection method of Appendix A of Dimitrova, Jia, and Kaishev (2026b), which generalises the univariate algorithm of Section 3.1 to $k = 2$.

Bivariate empirical cdf computation. The evaluation of the bivariate empirical cdfs F_m and G_n at the required grid points is performed using the fast divide-and-conquer algorithm of Langren\'e and Warin (2021). For N data points in d dimensions this algorithm achieves $O\big(N \log(N)^{(d-1)\vee 1}\big)$ complexity, compared to the naive $O(N^2)$, by recursively splitting the dataset along alternating coordinate axes and accumulating counts in each sub-problem. In the bivariate case ($d = 2$) this gives $O(N \log^2 N)$ complexity.

Projection method (Appendix A, Dimitrova, Jia, and Kaishev 2026b). The algorithm proceeds in five steps.

Step 1 (omnidirectional jump vertices of F_m). A point $\alpha \in A_{x^{(1)}} \times A_{x^{(2)}}$ is an *omnidirectional jump* of F_m if and only if $F_m(\alpha) \neq F_m(\alpha - \varepsilon_0 e_i)$ for both $i = 1$ and $i = 2$ (Definition 2.11), where $e_1 = (1, 0)$, $e_2 = (0, 1)$, and $\varepsilon_0 = \frac{1}{2}\min_{i;j \neq l} |x_j^{(i)} - x_l^{(i)}|$. Each omnidirectional jump α_i gives rise to two vertices of the planar curve F_m^c (Equation (19)):

$$A_{2i-1} = (\alpha_i,\, F_m(\alpha_i - \varepsilon)), \qquad A_{2i} = (\alpha_i,\, F_m(\alpha_i)), \qquad i = 1, \ldots, v,$$

where $\varepsilon = (\varepsilon_0, \varepsilon_0)$.

Step 2 (truncation boundary and additional vertices). The region between the bivariate planar curves is unbounded. Lemma 2.3 is applied to truncate both curves at $M = 1 + \max\{x_j^{(i)}, y_l^{(i)}\}$, ensuring that $H(\hat{F}_m^c(M), \hat{G}_n^c(M)) = H(F_m^c, G_n^c)$.

Step 3 (projection of F_m^c vertices). Each vertex A_l of $\hat{F}_m^c(M)$ is projected onto the $x^{(1)}Ox^{(2)}$ plane along the direction $E_0 = (1, 1, -1) \in R^3$.

Step 4 (projection partition for G_n^c). The vertices and sides of G_n^c are projected in the same direction E_0, forming a partition of R^2 whose regions determine which coordinate of A_l drives the distance to G_n^c.

Step 5 (nearest-point distances via Lemma A.3). For each locally farthest vertex $A_l \in \mathcal{V}_{loc}$, the nearest point on G_n^c is found via the Pareto frontier of dominated projected vertices, and the closed-form distance formula of Lemma A.3 is applied. The Hausdorff distance is

$$H(F_m^c, G_n^c) = \max_{A_l \in \mathcal{V}_{loc}} \inf_{B \in G_n^c} \rho_1(A_l, B).$$

The combinatorial search is executed by the internal C++ routine `hsearch_Rcpp` (via **Rcpp**).

Component-wise orders and $H_{m,n}^{\circ}$. The function computes the statistic under the standard component-wise order $\preceq_1$ ($\boldsymbol{x} \preceq_1 \boldsymbol{y}$ iff $x^{(1)} \leq y^{(1)}$ and $x^{(2)} \leq y^{(2)}$). The order-invariant statistic $H_{m,n}^{\circ} = \max_{1 \leq i \leq 4} H(F_{m,i}^c, G_{n,i}^c)$ (Equation (17) of Dimitrova, Jia, and Kaishev 2026b) is available directly via `Hausdorff_stat` with `invariant = TRUE`, which calls this function over the four sign-flip orderings and takes the maximum.

The inputs `x` and `y` must each be two-column numeric matrices; otherwise the function stops with an error. To pass list input, use `Hausdorff_stat`, which converts lists to matrices before calling this function.

Value

A single numeric value: the Hausdorff distance $H(F_m^c, G_n^c)$ between the bivariate empirical cdfs of `x` and `y`.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted to the Annals of Statistics*.

Nicolas Langren\'e, Xavier Warin (2021). "Fast Multivariate Empirical Cumulative Distribution Function with Connection to Kernel Density Estimation". *Computational Statistics & Data Analysis*, **162**, 107267. doi:10.1016/j.csda.2021.107267.

See Also

`Hausdorff_stat`, `Hausdorff_test`.

Examples

```
## Hausdorff distance between two bivariate samples from the same distribution
set.seed(1)
x <- matrix(rnorm(100), ncol = 2)
y <- matrix(rnorm(100), ncol = 2)
```

```
H_stat_2s_2d(x, y)

## Hausdorff distance between two bivariate samples from different distributions
set.seed(2)
x2 <- matrix(rnorm(100), ncol = 2)
y2 <- matrix(rnorm(100, mean = 0.5), ncol = 2)
H_stat_2s_2d(x2, y2)

## List input: use Hausdorff_stat() which handles the conversion
set.seed(3)
x3 <- list(rnorm(30), rnorm(30))
y3 <- list(rnorm(30), rnorm(30))
Hausdorff_stat(x3, y3)

## Order-invariant statistic via the unified wrapper
Hausdorff_stat(x, y, invariant = TRUE)
```

`H_test_1s_1d` *One-sample Hausdorff goodness-of-fit test*

Description

A thin wrapper that first computes the one-sample Hausdorff test statistic $\hat{q} = H(F^c, F_n^c)$ via `H_stat_1s_1d`, then delegates the p-value computation to `H_test_c_cdf`. The split into two functions allows the p-value to be computed independently when $\hat{q}$ is already known.

Usage

```
H_test_1s_1d(x, CDF, CDFinverse = NULL, pdf = NULL, tol = 1e-10,
             max.init = 1000)
```

Arguments

`x`	a numeric vector of data sample values $\{x_1, \ldots, x_n\}$.
`CDF`	a vectorised R function for the null cdf $F(x)$.
`CDFinverse`	(optional) a vectorised R function for $F^{-1}(p)$, extended so that $F^{-1}(p) = -\infty$ for $p < 0$ and $F^{-1}(p) = +\infty$ for $p > 1$. When supplied, the boundary values a_i and b_i are computed directly in `H_test_c_cdf`. Defaults to `NULL`.
`pdf`	(optional) a vectorised R function for the null density $f(x)$. When supplied, Newton–Raphson is used in `H_stat_1s_1d` to solve the equation $x + F(x) = \lambda$. Defaults to `NULL`.
`tol`	tolerance for root-finding, passed to both `H_stat_1s_1d` and `H_test_c_cdf`. Defaults to `1e-10`.
`max.init`	maximum number of Newton–Raphson iterations, passed to `H_stat_1s_1d`. Used only when `pdf` is supplied. Defaults to `1000`.

Value

An object of class "htest" with the following components:

statistic the observed Hausdorff statistic $\hat{q} = H(F^c, F_n^c)$, named "H".

p.value the exact p-value $P(H(F^c, F_n^c) > \hat{q})$.

method the character string "One-sample Hausdorff test".

alternative the character string "two-sided".

data.name a character string giving the name of the data object.

See Also

H_stat_1s_1d, H_test_c_cdf, Hausdorff_test.

Examples

```
set.seed(1)
x <- rexp(50, rate = 1)
H_test_1s_1d(x,
             CDF        = function(t) pexp(t, rate = 1),
             CDFinverse = function(p) qexp(p, rate = 1),
             pdf        = function(t) dexp(t, rate = 1))
```

H_test_2s_1d *Two-sample Hausdorff goodness-of-fit test with exact or Monte Carlo permutation p-values*

Description

Tests the null hypothesis $H_0 : F(x) = G(x)$ for all x against the two-sided alternative $H_1 : F(x) \neq G(x)$ for at least one x, using the Hausdorff distance between the empirical cdfs of two independent univariate samples. The p-value is obtained either by exact enumeration of all $\frac{(m+n)!}{m!n!}$ permutations of the pooled sample or by Monte Carlo permutation. The test statistic is computed by H_stat_2s_1d_tr.

Usage

```
H_test_2s_1d(x1, x2, nboots = 2000, Exact = FALSE)
```

Arguments

x1	a numeric vector of data sample values $\{x_1, \ldots, x_m\}$.
x2	a numeric vector of data sample values $\{y_1, \ldots, y_n\}$.
nboots	a positive integer giving the number of Monte Carlo permutation replications, used only when Exact = FALSE. Defaults to 2000.
Exact	a logical value indicating whether an exact permutation p-value should be computed by enumerating all $\frac{(m+n)!}{m!n!}$ splits of the pooled sample. If $\frac{(m+n)!}{m!n!} > 2.147483647 \times 10^9$, the function automatically switches to Monte Carlo and emits a message. Defaults to FALSE.

Details

Given two independent random samples $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$ from unknown univariate cdfs F and G respectively, the two-sample Hausdorff (H) test statistic is $\hat{q} = \mathcal{H}_{m,n} = H(F_m^c, G_n^c)$, computed by `H_stat_2s_1d_tr`. By Theorem 2.5 of Dimitrova, Jia, and Kaishev (2026b), this equals the side length of the largest axis-aligned square that can be inscribed in the region between the two empirical cdf staircase curves F_m^c and G_n^c.

Permutation p-value. Because the distribution of $\mathcal{H}_{m,n}$ depends on the unknown F and G, p-values are computed via a permutation argument (Proposition 3.5 of Dimitrova, Jia, and Kaishev 2026b). Let $Z_{m+n} = \{x_1, \ldots, x_m, y_1, \ldots, y_n\}$ be the pooled sample. The permutation Hausdorff statistic $\mathcal{H}_{m,n}^{\dagger}$ is defined on all $C = \frac{(m+n)!}{m!n!}$ splits of Z_{m+n} into groups of sizes m and n, and its conditional distribution given Z_{m+n} is:

$$P(\mathcal{H}_{m,n}^{\dagger} > q \mid Z_{m+n}) = \frac{1}{C} \sum_{i=1}^{C} \mathbf{1}\{H(F_m^{c\dagger}(\pi_i), G_n^{c\dagger}(\pi_i)) > q\},$$

where $F_m^{c\dagger}(\pi_i)$ and $G_n^{c\dagger}(\pi_i)$ are the empirical cdf planar curves of the i-th permuted sub-samples.

When `Exact = TRUE` all C splits are enumerated. When `Exact = FALSE`, the p-value is estimated by Monte Carlo: in each of `nboots` replications the pooled sample is randomly split into groups of sizes m and n and H is recomputed; the p-value is the proportion of replications for which $H \geq \hat{q}$. If the estimated p-value is zero it is replaced by $1/(2 \times \texttt{nboots})$.

Asymptotic equivalence (Theorems 3.2–3.4). Under the null hypothesis, or under fixed or contiguous alternatives (with bounded densities and $m/(m+n) \to \eta \in (0, 1)$), the permutation p-value is asymptotically equivalent to the true p-value of $\mathcal{H}_{m,n}$. The permutation test controls the type I error at the nominal level for all $k \geq 1$, and the power of the two tests are asymptotically equal.

Scale tuning. The power of $\mathcal{H}_{m,n}$ is not scale-invariant. The optimal scale σ^* can be estimated and applied automatically by passing `scale_psi` to the unified wrapper `Hausdorff_test`, which calls this function internally on the already-scaled samples.

Value

An object of class `"htest"` with the following components:

`statistic` the observed Hausdorff statistic $\hat{q} = \mathcal{H}_{m,n} = H(F_m^c, G_n^c)$, named `"H"`.

`p.value` the exact or Monte Carlo permutation p-value, i.e. the proportion of permutation statistics $H(F_m^{c\dagger}(\pi_i), G_n^{c\dagger}(\pi_i))$ that are greater than or equal to $\hat{q}$. If the Monte Carlo estimate is zero it is replaced by $1/(2 \times \texttt{nboots})$.

`method` a character string indicating the method: `"Two-sample Hausdorff Test (Exact)"` when all permutations are enumerated, or `"Two-sample Hausdorff Test (Monte Carlo)"` otherwise.

`alternative` the character string `"two-sided"`.

`data.name` a character string giving the names of the two data objects.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted to the Annals of Statistics*.

Dimitrina S. Dimitrova, Vladimir K. Kaishev, Senren Tan (2020). "Computing the Kolmogorov-Smirnov Distribution When the Underlying CDF is Purely Discrete, Mixed or Continuous". *Journal of Statistical Software*, **95**(10): 1–42. doi:10.18637/jss.v095.i10.

See Also

H_stat_2s_1d_tr, Hausdorff_test.

Examples

```
## Two-sample H test: both samples from N(0,1) (null is true)
set.seed(1)
x1 <- rnorm(30)
x2 <- rnorm(30)
H_test_2s_1d(x1, x2, nboots = 1000)

## Two-sample H test: samples from N(0,1) and N(0.5,1) (null is false)
set.seed(2)
x3 <- rnorm(30)
x4 <- rnorm(30, mean = 0.5)
H_test_2s_1d(x3, x4, nboots = 1000)

## Exact permutation test for small samples
set.seed(3)
a <- rnorm(8)
b <- rnorm(8)
H_test_2s_1d(a, b, Exact = TRUE)

## Scale-tuned test via the unified wrapper (recommended)
set.seed(4)
x_exp <- rexp(60, rate = 2)
y_exp <- rexp(60, rate = 3)
Hausdorff_test(x_exp, y_exp, scale_psi = c(0.99, 0.95), scale_nperms = 500)
```

H_test_2s_2d — *Two-sample bivariate Hausdorff goodness-of-fit test with Monte Carlo permutation p-values*

Description

Tests the null hypothesis $H_0 : F(\boldsymbol{x}) = G(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^2$ against the two-sided alternative $H_1 : F(\boldsymbol{x}) \neq G(\boldsymbol{x})$ for at least one $\boldsymbol{x}$, using the Hausdorff distance between the bivariate empirical cdfs of two independent samples. The p-value is obtained by Monte Carlo permutation. The test statistic is computed by H_stat_2s_2d.

Usage

```
H_test_2s_2d(x, y, nboots = 2000, Exact = FALSE, invariant = FALSE, tol = 1e-6)
```

Arguments

x	a two-column numeric matrix representing the first bivariate sample of size m.
y	a two-column numeric matrix representing the second bivariate sample of size n.
nboots	a positive integer giving the number of Monte Carlo permutation replications, used only when Exact = FALSE. Defaults to 2000.
Exact	a logical value indicating whether an exact permutation p-value should be computed by enumerating all $\frac{(m+n)!}{m!\,n!}$ splits of the pooled sample. If $\frac{(m+n)!}{m!\,n!} > 2.147483647 \times 10^9$ the function automatically switches to Monte Carlo and emits a message. Defaults to FALSE.
invariant	logical. When TRUE, the order-invariant statistic $H^{\circ}_{m,n}$ is used for the observed value and all permutation replicates, maximising over the four sign-flip orderings of the two coordinates (Equation (17) of Dimitrova, Jia, and Kaishev 2026b). Defaults to FALSE.
tol	a numeric value giving the tolerance for locating omnidirectional jump vertices in the bivariate empirical cdfs, passed to H_stat_2s_2d. Defaults to 1e-6.

Details

Given two independent random samples $\boldsymbol{X} = \{(X_{1,1}, X_{1,2}), \ldots, (X_{m,1}, X_{m,2})\}$ and $\boldsymbol{Y} = \{(Y_{1,1}, Y_{1,2}), \ldots, (Y_{n,1}, Y_{n,2})\}$ from unknown bivariate cdfs F and G respectively, the two-sample Hausdorff (H) test statistic is $\hat{q} = \mathcal{H}_{m,n} = H(F^c_m, G^c_n)$, computed by H_stat_2s_2d. By Theorem 2.5 of Dimitrova, Jia, and Kaishev (2026b), this equals the side length of the largest axis-aligned hypercube that can be inscribed in the region between the two bivariate empirical cdf surfaces F^c_m and G^c_n.

Permutation p-value. Because the null distribution of $\mathcal{H}_{m,n}$ depends on the unknown F and G, p-values are obtained by permutation (Proposition 3.5 of Dimitrova, Jia, and Kaishev 2026b). When Exact = TRUE all $\frac{(m+n)!}{m!\,n!}$ splits of the pooled sample are enumerated; if the count exceeds 2.147×10^9 the function falls back to Monte Carlo automatically. When Exact = FALSE the pooled sample is randomly split for nboots replications and the p-value is the proportion of permutation statistics $\geq \hat{q}$, floored at $1/(2 \times \text{nboots})$. The asymptotic equivalence between the permutation p-value and the true p-value of $\mathcal{H}_{m,n}$, and the control of type I error, follow from Theorems 3.2–3.4 of Dimitrova, Jia, and Kaishev (2026b); see H_test_2s_1d for the full statement.

Order-invariant statistic. When invariant = TRUE the order-invariant statistic $H^{\circ}_{m,n} = \max_{1 \leq i \leq 4} H(F^c_{m,i}, G^c_{n,i})$ (Equation (17) of Dimitrova, Jia, and Kaishev 2026b) is used for both the observed value and every permutation replicate. It maximises over the $2^2 = 4$ sign-flip orderings of the two coordinates, making the test invariant to the labelling of coordinate axes.

Scale tuning. For scale-tuned testing pass scale_psi to Hausdorff_test, which estimates the column-wise optimal scale vector (σ^*_1, σ^*_2) and calls this function internally on the already-scaled samples.

Value

An object of class "htest" with the following components:

statistic the observed Hausdorff statistic $\hat{q} = \mathcal{H}_{m,n} = H(F^c_m, G^c_n)$ (or $H^{\circ}_{m,n}$ when invariant = TRUE), named "H".

`p.value` the Monte Carlo permutation p-value, i.e. the proportion of permutation statistics greater than or equal to $\hat{q}$. If zero, replaced by $1/(2 \times \texttt{nboots})$.

`method` a character string indicating the procedure: `"Two-sample bivariate Hausdorff test (Exact)"` or `"Two-sample bivariate Hausdorff test (Monte Carlo permutation)"`, with the suffix `", order-invariant"` appended when `invariant = TRUE`.

`alternative` the character string `"two-sided"`.

`data.name` a character string giving the names of the two data objects.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026b). "On a Two-Sample Multivariate Goodness-of-Fit Test Based on the Hausdorff Metric". *Submitted to the Annals of Statistics*.

See Also

`H_stat_2s_2d`, `Hausdorff_test`, `H_test_2s_1d`.

Examples

```
## Bivariate H test: both samples from N(0,I) (null is true)
set.seed(1)
xm <- matrix(rnorm(100), ncol = 2)
ym <- matrix(rnorm(100), ncol = 2)
H_test_2s_2d(xm, ym, nboots = 1000)

## Bivariate H test: samples from N(0,I) and N(0.5,I) (null is false)
set.seed(2)
xm2 <- matrix(rnorm(100), ncol = 2)
ym2 <- matrix(rnorm(100, mean = 0.5), ncol = 2)
H_test_2s_2d(xm2, ym2, nboots = 1000)

## Exact permutation test for small samples
set.seed(3)
H_test_2s_2d(matrix(rnorm(16), ncol = 2),
             matrix(rnorm(16), ncol = 2), Exact = TRUE)

## Order-invariant statistic
set.seed(4)
H_test_2s_2d(xm, ym, nboots = 1000, invariant = TRUE)

## Scale-tuned test via the unified wrapper (recommended)
set.seed(4)
H_test_2s_2d(xm, ym, nboots = 1000)
Hausdorff_test(xm, ym, nboots = 1000,
               scale_psi = c(0.99, 0.95), scale_nperms = 500)
```

`H_test_c_cdf` *Exact complementary cdf of the one-sample Hausdorff statistic*

Description

Computes the exact complementary cdf $P(H(F^c, F_n^c) > q)$ of the one-sample Hausdorff goodness-of-fit statistic for a sample of size n. For a continuous null cdf F, the rectangle-probability representation of Theorem 4 of Dimitrova, Jia, and Kaishev (2026a) shows that

$$P(H(F^c, F_n^c) > q) = 1 - P(a_i \le U_{(i)} \le b_i,\ 1 \le i \le n),$$

where $U_{(i)}$ are the order statistics of an i.i.d.\ $U(0,1)$ sample and

$$a_i = F\left(F^{-1}\left(\frac{i}{n} - q\right) - q\right), \qquad b_i = F\left(F^{-1}\left(\frac{i-1}{n} + q\right) + q\right).$$

The function evaluates this rectangle-probability representation numerically via `KSgeneral::ks_c_cdf_Rcpp()` after constructing the boundary vectors required by that routine.

This is the low-level p-value engine used by `H_test_1s_1d` and is useful when the observed Hausdorff statistic q has already been computed.

Usage

```
H_test_c_cdf(q, n, CDF, CDFinverse = NULL, pdf = NULL, tol = 1e-10,
             max.init = 1000)
```

Arguments

`q`	a single numeric value giving the observed Hausdorff statistic.
`n`	a positive integer, the sample size.
`CDF`	a vectorised R function implementing the null cdf $F(x)$.
`CDFinverse`	(optional) a vectorised R function implementing $F^{-1}(p)$. When supplied, the boundary values are computed directly: $$a_i = F\left(F^{-1}\left(\frac{i}{n} - q\right) - q\right), \qquad b_i = F\left(F^{-1}\left(\frac{i-1}{n} + q\right) + q\right),$$ where F^{-1} is understood in the extended sense with $F^{-1}(y) = -\infty$ for $y \le 0$ and $F^{-1}(y) = +\infty$ for $y \ge 1$. Defaults to `NULL`.
`pdf`	(optional) a vectorised R function for the null density $f(x)$. Used only when `CDFinverse` is `NULL`: if `pdf` is supplied, the required quantiles are obtained by Newton–Raphson iteration; otherwise by `uniroot` with `extendInt = "yes"`. Defaults to `NULL`.
`tol`	numerical tolerance used in the root-finding step. Defaults to `1e-10`.
`max.init`	maximum number of Newton–Raphson iterations allowed at each boundary point. Defaults to `1000`.

Details

For $q \in (0,1)$, the function constructs two boundary vectors `f_a` and `f_b` of length n. If `CDFinverse` is available, these are computed directly from the closed-form expressions for a_i and b_i. Otherwise, the code solves $F(x) = i/n - q$ and $F(x) = (i-1)/n + q$ numerically and maps the resulting roots through $F(x-q)$ and $F(x+q)$, respectively. When $i/n - q < 0$ the entry $a_i = 0$ is assigned directly; when $(i-1)/n + q > 1$ the entry $b_i = 1$ is assigned directly, avoiding unnecessary root-finding.

The two boundary vectors are written to a temporary file '`Boundary_Crossing_Time.txt`', then passed to `KSgeneral::ks_c_cdf_Rcpp(n)`, which evaluates the required rectangle probability in $O(n^2 \log n)$ time. The file is deleted after use.

The trivial boundary cases `q >= 1` (returns 1) and `q <= 0` (returns 0) are handled analytically before any computation.

Value

A single numeric value equal to the exact complementary cdf $P(H(F^c, F_n^c) > q)$.

References

Dimitrina S. Dimitrova, Yun Jia, Vladimir K. Kaishev (2026a). "On a One Sample Goodness-of-fit Test Based on the Hausdorff Metric". *Submitted*.

Dimitrina S. Dimitrova, Vladimir K. Kaishev, Senren Tan (2020). "Computing the Kolmogorov-Smirnov Distribution When the Underlying CDF is Purely Discrete, Mixed or Continuous". *Journal of Statistical Software*, **95**(10), 1–42. doi:10.18637/jss.v095.i10.

See Also

`H_test_1s_1d`, `H_stat_1s_1d`, `Hausdorff_test`.

Examples

```
set.seed(1)
x <- rexp(50, rate = 1)
q <- H_stat_1s_1d(x   = x,
                  CDF = function(t) pexp(t, rate = 1),
                  pdf = function(t) dexp(t, rate = 1))
H_test_c_cdf(q          = q,
             n          = length(x),
             CDF        = function(t) pexp(t, rate = 1),
             CDFinverse = function(p) qexp(p, rate = 1),
             pdf        = function(t) dexp(t, rate = 1))
```

Index