Package 'fasano.franceschini.test'

August 17, 2022

Type Package

Title Fasano-Franceschini Test: A Multidimensional Kolmogorov-Smirnov Two-Sample Test

Version 2.0.1

Description An implementation of the two-sample multidimensional Kolmogorov-Smirnov test described by Fasano and Franceschini (1987) <doi:10.1093/mnras/225.1.155>. This test evaluates the null hypothesis that two i.i.d. random samples were drawn from the same underlying probability distribution. The data can be of any dimension, and can be of any type (continuous, discrete, or mixed).

License MIT + file LICENSE

URL https://github.com/nesscoder/fasano.franceschini.test

BugReports https://github.com/nesscoder/fasano.franceschini.test/issues

Depends R (>= 3.0.2)

Imports microbenchmark, Rcpp (>= 1.0.0), RcppParallel (>= 5.0.1), stats

Suggests testthat (>= 3.0.0)

LinkingTo Rcpp (>= 1.0.0), RcppParallel (>= 5.0.1)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.1

SystemRequirements GNU make

NeedsCompilation yes

Author Connor Puritz [aut, cre] (<https://orcid.org/0000-0001-7602-0444>), Elan Ness-Cohn [aut] (<https://orcid.org/0000-0002-3935-6667>), Rosemary Braun [ctb, ths] (<https://orcid.org/0000-0001-9668-9866>), Luca Weihs [cph] (Copyright holder and author of 'RangeTree' class.)

Maintainer Connor Puritz <connorpuritz2025@u.northwestern.edu>

Repository CRAN

Date/Publication 2022-08-17 19:30:02 UTC

R topics documented:

`fasano.franceschini.test`
Fasano-Franceschini Test

Description

Performs a two-sample multidimensional Kolmogorov-Smirnov test as described by Fasano and Franceschini (1987). This test evaluates the null hypothesis that two i.i.d. random samples were drawn from the same underlying probability distribution. The data can be of any dimension, and can be of any type (continuous, discrete, or mixed).

Usage

```
fasano.franceschini.test(
  S1,
  S2,
  nPermute = 100,
  threads = 1,
  cores,
  seed = NULL,
  p.conf.level = 0.95,
  verbose = TRUE,
  method = c("o", "r", "b")
)
```

Arguments

`S1`	`matrix` or `data.frame`.
`S2`	`matrix` or `data.frame`.
`nPermute`	A nonnegative `integer` setting the number of permuted samples to generate when estimating the permutation test p-value. Default is `100`. If set to `0`, no p-value is estimated.
`threads`	A positive `integer` or `"auto"` setting the number of threads used for performing the permutation test. If set to `"auto"`, the number of threads is determined by `RcppParallel::defaultNumThreads()`. Default is `1`.
`cores`	Allowed for backwards compatibility. `threads` is now the preferred argument name.
`seed`	Optional integer to seed the PRNG used for the permutation test. Default is `NULL`. Only available for serial version (`threads = 1`).
`p.conf.level`	Confidence level for the confidence interval of the permutation test p-value.

`verbose` A `boolean` indicating whether to display a progress bar. Default is `TRUE`. Only available for serial version (`threads = 1`).

`method` A `character` indicating which method to use to compute the test statistic. All methods return the same results but may vary in computation speed. The options are:

- `'o'` - Optimize: selects the fastest method for the given data.
- `'r'` - Range tree
- `'b'` - Brute force

See the Details section for more information about each method.

Details

The test statistic can be computed using two different methods. Both methods return identical results, but have different time complexities:

- Range tree method: This method has a time complexity of *O(n*log(n)^(d-1))*, where *n* is the size of the larger sample and *d* is the dimension of the data.
- Brute force method: This method has a time complexity of *O(n^2)*.

The range tree method tends to be faster for low dimensional data or large sample sizes, while the brute force method tends to be faster for high dimensional data or small sample sizes. When `method` is set to `` `o` ``, the test statistic is computed once using each method, and whichever method is faster is used for the permutation test. To perform more comprehensive benchmarking, `nPermute` can be set equal to `0`, which bypasses the permutation test and only computes the test statistic.

The p-value for the test is computed empirically using a permutation test. As it is almost always infeasible to compute the exact permutation test p-value, a Monte Carlo approximation is made instead. This estimate is a binomially distributed random variable, and thus a confidence interval can be computed. The confidence interval is obtained using the procedure given in Clopper and Pearson (1934).

Value

A list with class `htest` containing the following components:

`statistic` The value of the test statistic *Z*.

`estimate` The value of the difference statistics *D1* and *D2*.

`p.value` The permutation test p-value.

`conf.int` A binomial confidence interval for the p-value.

`method` A character string indicating what type of test was performed.

`data.name` A character string giving the names of the data.

References

- Fasano, G. & Franceschini, A. (1987). A multidimensional version of the Kolmogorov-Smirnov test. *Monthly Notices of the Royal Astronomical Society*, 225:155-170. doi:10.1093/mnras/225.1.155.
- Clopper, C. J. & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26, 404–413. doi:10.2307/2331986.

Examples

```
set.seed(0)

# create 2-D samples
S1 <- data.frame(x = rnorm(n = 20, mean = 0, sd = 1),
                 y = rnorm(n = 20, mean = 1, sd = 2))
S2 <- data.frame(x = rnorm(n = 40, mean = 0, sd = 1),
                 y = rnorm(n = 40, mean = 1, sd = 2))

# perform test (serial version)
fasano.franceschini.test(S1, S2)

# perform test with more permutations
fasano.franceschini.test(S1, S2, nPermute = 150)

# set seed for reproducible p-value
fasano.franceschini.test(S1, S2, seed = 0)$p.value
fasano.franceschini.test(S1, S2, seed = 0)$p.value

# change confidence level for p-value confidence interval
fasano.franceschini.test(S1, S2, p.conf.level = 0.99)

# perform test using range tree method
fasano.franceschini.test(S1, S2, method = 'r')

# perform test using brute force method
fasano.franceschini.test(S1, S2, method = 'b')

# perform test (parallel version, 2 threads)
## Not run:
fasano.franceschini.test(S1, S2, threads = 2)

## End(Not run)
```

Index