Package 'twosamples'

December 3, 2018

Type Package

Title Fast Permutation Based Two Sample Tests

Version 1.0.0

Author Connor Dowd

Maintainer Connor Dowd <cdowd@chicagobooth.edu>

Description Fast randomization based two sample tests.
Testing the hypothesis that two samples come from the same distribution
using randomization to create p-values. Included tests are: Kolmogorov-Smirnov,
Kuiper, Cramer-von Mises, and Anderson-Darling. There is also a very efficient test
based on the Wasserstein Distance. The default test 'two_sample' builds on the
Wasserstein distance by using a weighting scheme like that of Anderson-Darling.
We also include the permutation scheme to make test building simple for others.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.17)

LinkingTo Rcpp

RoxygenNote 6.1.1

URL https://github.com/cdowd/twosamples

BugReports https://github.com/cdowd/twosamples/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-12-03 09:30:03 UTC

R topics documented:

`ad_stat` *Anderson-Darling Test*

Description

Anderson-Darling Test

Usage

```
ad_stat(a, b, power = 2)

ad_test(a, b, nboots = 2000, p = default.p)
```

Arguments

`a`	a vector of numbers
`b`	a vector of numbers
`power`	power to raise test stat to
`nboots`	Number of bootstrap iterations
`p`	power to raise test stat to

Details

The AD test compares two ECDFs by looking at the weighted sum of the squared differences between them – evaluated at each point in the joint sample. The weights are determined by the variance of the joint ECDF at that point. Formally – if E is the ECDF of sample 1, F is the ECDF of sample 2, and G is the ECDF of the joint sample then CVM = SUM_(x in k) (E(x)-F(x))^2/(G(x)*(1-G(x))) where k is the joint sample. The test p-value is calculated by randomly resampling two samples of the same size using the combined sample. Intuitively the AD test improves on the CVM test by giving lower weight to noisy observations.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- `ad_stat`: Anderson-Darling Test statistic
- `ad_test`: Permutation based two sample Anderson-Darling test

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
ad_test(vec1,vec2)
```

`cvm_stat` *Cramer-vonMises Test*

Description

Cramer-vonMises Test

Usage

```
cvm_stat(a, b, power = 2)

cvm_test(a, b, nboots = 2000, p = default.p)
```

Arguments

`a`	a vector of numbers
`b`	a vector of numbers
`power`	power to raise test stat to
`nboots`	Number of bootstrap iterations
`p`	power to raise test stat to

Details

The CVM test compares two ECDFs by looking at the sum of the squared differences between them – evaluated at each point in the joint sample. Formally – if E is the ECDF of sample 1 and F is the ECDF of sample 2, then CVM = SUM_(x in k) (E(x)-F(x))^2 where k is the joint sample. The test p-value is calculated by randomly resampling two samples of the same size using the combined sample. Intuitively the CVM test improves on KS by using the full joint sample, rather than just the maximum distance – this gives it greater power against shifts in higher moments, like variance changes.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- `cvm_stat`: Cramer-Von Mises Test statistic
- `cvm_test`: Permutation based two sample Cramer-Von Mises test

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
cvm_test(vec1,vec2)
```

`dts_stat` *Two Sample Test*

Description

Two Sample Test

Usage

```
dts_stat(a, b, power = 1)

dts_test(a, b, nboots = 2000, p = default.p)

two_sample(a, b, nboots = 2000, p = default.p)
```

Arguments

`a`	a vector of numbers
`b`	a vector of numbers
`power`	also the power to raise the test stat to
`nboots`	Number of bootstrap iterations
`p`	power to raise test stat to

Details

The DTS test compares two ECDFs by looking at the reweighted Wasserstein distance between the two. If the wass_test extends cvm_test to interval data, then DTS extends ad_test to interval data. Formally – if E is the ECDF of sample 1, F is the ECDF of sample 2, and G is the ECDF of the combined sample, then DTS = Integral |E(x)-F(x)|/(G(x)(1-G(x))) across all x. The test p-value is calculated by randomly resampling two samples of the same size using the combined sample. Intuitively the DTS test improves on AD by allowing more extreme observations to carry more weight. At a higher level – CVM/AD/KS/etc only require ordinal data. DTS gains its power because it takes advantages of the properties of interval data – i.e. the distances have some meaning. This is the same argument as Wasserstein vs AD/CVM/KS. However, DTS, like Anderson-Darling (AD) also downweights noisier observations relative to WASS, thus (hopefully) giving it extra power.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- `dts_stat`: Test statistic based on a weighted area between ECDFs
- `dts_test`: Permutation based two sample test
- `two_sample`: Recommended two-sample test

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
dts_stat(vec1,vec2)
dts_test(vec1,vec2)
two_sample(vec1,vec2)
```

`ks_stat` *Kolmogorov-Smirnov Test*

Description

Performs a permutation based two sample test using the Kolmogorov-Smirnov test statistic (ks_stat).

Usage

```
ks_stat(a, b, power = 1)

ks_test(a, b, nboots = 2000, p = default.p)
```

Arguments

`a`	a vector of numbers
`b`	a vector of numbers
`power`	power to raise test stat to
`nboots`	Number of bootstrap iterations
`p`	power to raise test stat to

Details

The KS test compares two ECDFs by looking at the maximum difference between them. Formally – if E is the ECDF of sample 1 and F is the ECDF of sample 2, then KS = max |E(x)-F(x)| for values of x in the joint sample. The test p-value is calculated by randomly resampling two samples of the same size using the combined sample.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- ks_stat: Kolmogorov-Smirnov test statistic
- ks_test: Permutation based two sample Kolmogorov-Smirnov test

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
ks_test(vec1,vec2)
```

---

kuiper_stat *Kuiper Test*

Description

Performs a permutation based two sample test using the Kuiper test statistic (kuiper_stat).

Usage

```
kuiper_stat(a, b, power = 1)

kuiper_test(a, b, nboots = 2000, p = default.p)
```

Arguments

a	a vector of numbers
b	a vector of numbers
power	power to raise test stat to
nboots	Number of bootstrap iterations
p	power to raise test stat to

Details

The Kuiper test compares two ECDFs by looking at the maximum positive and negative difference between them. Formally – if E is the ECDF of sample 1 and F is the ECDF of sample 2, then KUIPER = max_x E(x)-F(x) + max_x F(x)-E(x). The test p-value is calculated by randomly resampling two samples of the same size using the combined sample.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- kuiper_stat: Kuiper Test statistic
- kuiper_test: Permutation based two sample Kuiper test

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
kuiper_test(vec1,vec2)
```

---

order_cpp *Order in C++*

Description

Simply finds the order of a vector in c++. Mostly for internals.

Usage

```
order_cpp(x)
```

Arguments

x	numeric vector

Value

same length vector of integers representing order of input vector

Examples

```
vec = c(1,4,3,2)
order_cpp(vec)
```

---

permutation_test_builder
Permutation Test Builder

Description

This function takes a simple two-sample test statistic and produces a function which performs permutation tests using that test stat.

Usage

```
permutation_test_builder(test_stat_function, default.p = 2)
```

Arguments

test_stat_function	a function of two vectors producing a positive number, intended as the test-statistic to be used.
default.p	This allows for some introduction of defaults and parameters. Typically used to control the power functions raise something to.

Details

test_stat_function must be structured to take two separate vectors, and then a third value. i.e. (fun = function(vec1,vec2,val1) ...). See examples.

Value

This function returns a function which will perform permutation tests on given test stat.

Functions

- permutation_test_builder: Takes a test statistic, returns a testing function.

Examples

```
mean_stat = function(a,b,p) abs(mean(a)-mean(b))**p
myfun = permutation_test_builder(mean_stat,2.0)
vec1 = rnorm(20)
vec2 = rnorm(20,4)
myfun(vec1,vec2)
```

---

wass_stat *Wasserstein Distance Test A two-sample test based on Wasserstein's distance.*

Description

Wasserstein Distance Test A two-sample test based on Wasserstein's distance.

Usage

```
wass_stat(a, b, power = 1)

wass_test(a, b, nboots = 2000, p = default.p)
```

Arguments

a	a vector of numbers
b	a vector of numbers
power	power to raise test stat to
nboots	Number of bootstrap iterations
p	power to raise test stat to

Details

The Wasserstein test compares two ECDFs by looking at the Wasserstein distance between the two. This is of course the area between the two ECDFs. Formally – if E is the ECDF of sample 1 and F is the ECDF of sample 2, then WASS = Integral |E(x)-F(x)| across all x. The test p-value is calculated by randomly resampling two samples of the same size using the combined sample. Intuitively the Wasserstein test improves on CVM by allowing more extreme observations to carry more weight. At a higher level – CVM/AD/KS/etc only require ordinal data. Wasserstein gains its power because it takes advantages of the properties of interval data – i.e. the distances have some meaning.

Value

Output is a length 2 Vector with test stat and p-value in that order. That vector has 3 attributes – the sample sizes of each sample, and the number of bootstraps performed for the pvalue.

Functions

- `wass_stat`: Wasserstein metric between two ECDFs
- `wass_test`: Permutation based two sample test using Wasserstein metric

Examples

```
vec1 = rnorm(20)
vec2 = rnorm(20,4)
wass_test(vec1,vec2)
```

Index