Package 'EDFtest'

January 20, 2025

Type Package

Title Goodness of Fit Based on Empirical Distribution Function

Version 0.1.0

Date 2021-10-22

Description
This repository contains software for the calculation of goodness-of-fit test statistics and their P-values. The three statistics computed are the Empirical Distribution function statistics called Cramer-von Mises, Anderson-Darling, and Watson statistics. The statistics and their P-values can be used to assess an assumed distribution.The following distributions are available: Uniform, Normal, Gamma, Logistic, Laplace, Weibull, Extreme Value, and Exponential.

License MIT + file LICENSE

Encoding UTF-8

BugReports `https://github.com/LiYao-sfu/EDFtest/issues`

RoxygenNote 7.1.1

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, stringi, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports CompQuadForm (>= 1.4.3), rmutil (>= 1.1.5), stats

VignetteBuilder knitr

NeedsCompilation no

Author Li Yao [aut, cre],
Richard Lockhart [aut]

Maintainer Li Yao <`yaoliy@sfu.ca`>

Repository CRAN

Date/Publication 2021-10-25 07:20:14 UTC

Contents

`EDFtest-package` *EDFtest: Goodness-of-fit Tests based on Empirical Distribution Function*

Description

`EDFtest` provides software for the calculation of goodness-of-fit test statistics and their P-values. The three statistics computed are the Empirical Distribution function statistics called Cramér-von Mises, Anderson-Darling, and Watson statistic.

Author(s)

Maintainer: Li Yao <`yaoliy@sfu.ca`>

Authors:

- Richard Lockhart
- Li Yao

References

Stephens, M.A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons. Journal of the American Statistical Association, Vol. 69, 730-737. Available at doi: 10.2307/2286009.

Stephens, M.A. (1976). Asymptotic results for goodness-of-fit statistics with unknown parameters. Annals of Statistics, Vol. 4, 357-369. Available at `https://scholar.google.com/scholar_url?url=https://projecteuclid.org/journals/annals-of-statistics/volume-4/issue-2/Asymptotic-Results-fo`
`10.1214/aos/1176343411.pdf&hl=en&sa=T&oi=ucasa&ct=ufr&ei=Wmd0YePQH_iM6rQPgq2vuAg&`
`scisig=AAGBfm09GbS_cTMrgCfM7KEYz6yjOKMUfQ`

See Also

Useful links:

- https://github.com/LiYao-sfu/EDFtest
- Report bugs at https://github.com/LiYao-sfu/EDFtest/issues

`AD` *Anderson-Darling statistic*

Description

Compute the Anderson-Darling goodness-of-fit statistic A^2 for an i.i.d sample, x, to test for the given distribution with parameters unknown.Estimate parameters by ML using `EDFtest` MLE function by default.

Usage

```
AD.uniform(x, parameter = estimate.uniform(x))

AD.normal(x, parameter = estimate.normal(x))

AD.gamma(x, parameter = estimate.gamma(x))

AD.logistic(x, parameter = estimate.logistic(x))

AD.laplace(x, parameter = estimate.laplace(x))

AD.weibull(x, parameter = estimate.weibull(x))

AD.extremevalue(x)

AD.exp(x, parameter = estimate.exp(x))

AD(z)
```

Arguments

`x`	A random sample.
`parameter`	Parameters of the given distribution, MLE by default.
`z`	A standard uniform random sample.

Value

Anderson-Darling statistic of the given sample.

See Also

estimate for estimating distribution parameters by ML; CvM for calculating Cramér-von Mises statistic; Watson for calculating Watson statistic; AD.pvalue for calculating P-value of Anderson-Darling statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
AD.uniform(x0)

x1=rnorm(n=100,mean=0,sd=1)
AD.normal(x1)

x2=rgamma(n=100,shape=1,scale=1)
AD.gamma(x2)

x3=rlogis(n=100,location=0,scale=1)
AD.logistic(x3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
AD.laplace(x4)

x5=rweibull(n=100,shape=1,scale=1)
AD.weibull(x5)
x5_log=log(x5)
AD.extremevalue(x5_log)

x6=rexp(n=100,rate=1/2)
AD.exp(x6)
```

AD.pvalue *P-value of Anderson-Darling statistic*

Description

Compute the P-value of the given Anderson-Darling statistic A^2 using imhof function in CompQuadForm.

Usage

```
AD.uniform.pvalue(a, neig = 100, verbose = FALSE)

AD.normal.pvalue(a, neig = 100, verbose = FALSE)

AD.gamma.pvalue(a, shape, neig = 100, verbose = FALSE)

AD.logistic.pvalue(a, neig = 100, verbose = FALSE)

AD.laplace.pvalue(a, neig = 100, verbose = FALSE)
AD.weibull.pvalue(a, neig = 100, verbose = FALSE)

AD.extremevalue.pvalue(a, neig = 100, verbose = FALSE)

AD.exp.pvalue(a, neig = 100, verbose = FALSE)
```

Arguments

`a`	Anderson-Darling statistic A^2 with a given distribution.
`neig`	Number of eigenvalues used for `imhof`.
`verbose`	Logical; if TRUE, print warning messages.
`shape`	The shape parameter of Gamma distribution.

Details

Parameters must be estimated by maximum likelihood (ML) in order for the P-values computed here to be asymptotically valid. They are computed using the fact that when parameters are estimated by maximum likelihood and the null hypothesis is true, the asymptotic distribution of the GOF statistic is the distribution of an infinite weighted sum of weighted chi-square random variables on 1 degree of freedom. The weights are eigenvalues of an integral equation. They depend on the distribution being tested, the statistic being used, and in some cases on the actual parameter values. These weights are then computed approximately by discretization of the integral equation; when that equation depends on one or more parameter values we use the MLE in the equation.

Some notes on the specific distributions: For the Normal, Logistic, Laplace, Extreme Value, Weibull and Exponential distributions, the limiting distributions do not depend on the parameters. For the Gamma distribution, the shape parameter affects the limiting distribution. The tests remain asymptotically valid when the MLE is used to approximate the limit distribution.

The Exponential distribution is a special case of the Weibull and Gamma families arising when the shape is known to be 1. Knowing a parameter and therefore not estimating it affects the distribution of the test statistic and the functions provided for the Exponential distribution allow for this.

If a data set X_1,...,X_n follows the Weibull distribution then Y_1 = log(X_1), ... ,Y_n = log(X_n) follows the Extreme Value distribution and vice versa. The two procedures give identical test statistics and P-values, in principal.

Some of the models have more than one common parametrization. For the Exponential, Gamma, and Weibull distributions, some writers use a rate parameter and some use the scale parameter which is the inverse of the rate. Our code uses the scale parameter.

For the Laplace distribution, some writers use the density $f(x) = exp(-|x-\mu|/\beta)/(2\beta)$ in which β is a scale parameter. Others use the standard deviation $\sigma = \beta/\sqrt{2}$. Our code uses the scale parameter.

For the Uniform distribution, we offer code for the two parameter Uniform distribution on the range θ_1 to θ_2. These are estimated by the sample minimum and sample maximum. The probability integral transforms of the remaining n-2 points are then tested for uniformity on the range 0 to 1. This procedure is justified because the these probability integral transforms have exactly this distribution if the original data had a uniform distribution over any interval.

It is not unusual to test the hypothesis that a sample follows the standard uniform distribution on [0,1]. In this case the parameters *should not* be estimated. Instead use `AD(z)` or `CvM(z)` or `Watson(z)` to compute the statistic values and then get P-values from `AD.uniform.pvalue(a)` or `CvM.uniform.pvalue(w)` or `Watson.uniform.pvalue(u)` whichever is wanted.

Value

P-value of the Anderson-Darling statistic.

See Also

`AD` for calculating Anderson-Darling statistic; `CvM.pvalue` for calculating P-value of Cramér-von Mises statistic; `Watson.pvalue` for calculating P-value of Watson statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
asq0 = AD.uniform(x0)
AD.uniform.pvalue(asq0)

x1=rnorm(n=100,mean=0,sd=1)
asq1 = AD.normal(x1)
AD.normal.pvalue(asq1)

x2=rgamma(n=100,shape=1,scale=1)
asq2 = AD.gamma(x2)
AD.gamma.pvalue(asq2,1)

x3=rlogis(n=100,location=0,scale=1)
asq3 = AD.logistic(x3)
AD.logistic.pvalue(asq3)

x4=rmutil::rlaplace(n=100,m=0,s=1)
asq4 = AD.laplace(x4)
AD.laplace.pvalue(asq4)

x5=rweibull(n=100,shape=1,scale=1)
asq5 = AD.weibull(x5)
AD.weibull.pvalue(asq5)
x5_log=log(x5)
AD.extremevalue.pvalue(AD.extremevalue(x5_log))

x6=rexp(n=100,rate=1/2)
asq6 = AD.exp(x6)
AD.exp.pvalue(asq6)
```

CvM *Cramer-von Mises statistic*

Description

Compute the Cramér-von Mises goodness-of-fit statistic W^2 for an i.i.d. sample, x, to test for the given distribution with parameters unknown. Estimate parameters by ML using `EDFtest` MLE function by default.

Usage

```
CvM.uniform(x, parameter = estimate.uniform(x))

CvM.normal(x, parameter = estimate.normal(x))

CvM.gamma(x, parameter = estimate.gamma(x))

CvM.logistic(x, parameter = estimate.logistic(x))

CvM.laplace(x, parameter = estimate.laplace(x))

CvM.weibull(x, parameter = estimate.weibull(x))

CvM.extremevalue(x)

CvM.exp(x, parameter = estimate.exp(x))

CvM(z)
```

Arguments

`x`	A random sample.
`parameter`	Parameters of the given distribution, MLE by default.
`z`	A standard uniform random sample.

Value

Cramér-von Mises statistic of the given sample.

See Also

`estimate` for estimating distribution parameters by ML; `AD` for calculating Anderson-Darling statistic; `Watson` for calculating Watson statistic; `CvM.pvalue` for calculating P-value of Cramér-von Mises statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
CvM.uniform(x0)

x1=rnorm(n=100,mean=0,sd=1)
CvM.normal(x1)

x2=rgamma(n=100,shape=1,scale=1)
CvM.gamma(x2)

x3=rlogis(n=100,location=0,scale=1)
CvM.logistic(x3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
CvM.laplace(x4)

x5=rweibull(n=100,shape=1,scale=1)
CvM.weibull(x5)
x5_log=log(x5)
CvM.extremevalue(x5_log)

x6=rexp(n=100,rate=1/2)
CvM.exp(x6)
```

`CvM.pvalue` *P-value of Cramer-von Mises statistic*

Description

Compute the P-value of the given Cramér-von Mises statistic W^2 using `imhof` function in `CompQuadForm`.

Usage

```
CvM.uniform.pvalue(w, neig = 100, verbose = FALSE)

CvM.normal.pvalue(w, neig = 100, verbose = FALSE)

CvM.gamma.pvalue(w, shape, neig = 100, verbose = FALSE)

CvM.logistic.pvalue(w, neig = 100, verbose = FALSE)

CvM.laplace.pvalue(w, neig = 100, verbose = FALSE)

CvM.weibull.pvalue(w, neig = 100, verbose = FALSE)

CvM.extremevalue.pvalue(w, neig = 100, verbose = FALSE)

CvM.exp.pvalue(w, neig = 100, verbose = FALSE)
```

Arguments

`w`	Cramér-von Mises statistic W^2 with a given distribution.
`neig`	Number of eigenvalues used for `imhof`.
`verbose`	Logical; if TRUE, print warning messages.
`shape`	The shape parameter of Gamma distribution.

Details

Parameters must be estimated by maximum likelihood (ML) in order for the P-values computed here to be asymptotically valid. They are computed using the fact that when parameters are estimated by maximum likelihood and the null hypothesis is true, the asymptotic distribution of the GOF statistic is the distribution of an infinite weighted sum of weighted chi-square random variables on 1 degree of freedom. The weights are eigenvalues of an integral equation. They depend on the distribution being tested, the statistic being used, and in some cases on the actual parameter values. These weights are then computed approximately by discretization of the integral equation; when that equation depends on one or more parameter values we use the MLE in the equation.

Some notes on the specific distributions: For the Normal, Logistic, Laplace, Extreme Value, Weibull and Exponential distributions, the limiting distributions do not depend on the parameters. For the Gamma distribution, the shape parameter affects the limiting distribution. The tests remain asymptotically valid when the MLE is used to approximate the limit distribution.

The Exponential distribution is a special case of the Weibull and Gamma families arising when the shape is known to be 1. Knowing a parameter and therefore not estimating it affects the distribution of the test statistic and the functions provided for the Exponential distribution allow for this.

If a data set X_1,...,X_n follows the Weibull distribution then Y_1 = log(X_1), ... ,Y_n = log(X_n) follows the Extreme Value distribution and vice versa. The two procedures give identical test statistics and P-values, in principal.

Some of the models have more than one common parametrization. For the Exponential, Gamma, and Weibull distributions, some writers use a rate parameter and some use the scale parameter which is the inverse of the rate. Our code uses the scale parameter.

For the Laplace distribution, some writers use the density $f(x) = exp(-|x - \mu|/\beta)/(2\beta)$ in which β is a scale parameter. Others use the standard deviation $\sigma = \beta/\sqrt{2}$. Our code uses the scale parameter.

For the Uniform distribution, we offer code for the two parameter Uniform distribution on the range θ_1 to θ_2. These are estimated by the sample minimum and sample maximum. The probability integral transforms of the remaining n-2 points are then tested for uniformity on the range 0 to 1. This procedure is justified because the these probability integral transforms have exactly this distribution if the original data had a uniform distribution over any interval.

It is not unusual to test the hypothesis that a sample follows the standard uniform distribution on [0,1]. In this case the parameters *should not* be estimated. Instead use `AD(z)` or `CvM(z)` or `Watson(z)` to compute the statistic values and then get P-values from `AD.uniform.pvalue(a)` or `CvM.uniform.pvalue(w)` or `Watson.uniform.pvalue(u)` whichever is wanted.

Value

P-value of the given Cramér-von Mises statistic.

See Also

CvM for calculating Cramér-von Mises statistic; AD.pvalue for calculating P-value of Anderson-Darling statistic; Watson.pvalue for calculating P-value of Watson statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
wsq0 = CvM.uniform(x0)
CvM.uniform.pvalue(wsq0)

x1=rnorm(n=100,mean=0,sd=1)
wsq1 = CvM.normal(x1)
CvM.normal.pvalue(wsq1)

x2=rgamma(n=100,shape=1,scale=1)
wsq2 = CvM.gamma(x2)
CvM.gamma.pvalue(wsq2,1)

x3=rlogis(n=100,location=0,scale=1)
wsq3 = CvM.logistic(x3)
CvM.logistic.pvalue(wsq3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
wsq4 = CvM.laplace(x4)
CvM.laplace.pvalue(wsq4)

x5=rweibull(n=100,shape=1,scale=1)
wsq5 = CvM.weibull(x5)
CvM.weibull.pvalue(wsq5)
x5_log=log(x5)
CvM.extremevalue.pvalue(CvM.extremevalue(x5_log))

x6=rexp(n=100,rate=1/2)
wsq6 = CvM.exp(x6)
CvM.exp.pvalue(wsq6)
```

`estimate` *MLE for univariate sample*

Description

Estimate parameters of various distributions by the method of maximum likelihood. The following families are available: Normal(location=μ,scale=σ^2), Gamma(shape=α,scale=β), Logistic(location=μ,scale=s), Laplace(location=μ,scale=b), Weibull(shape=α,scale=β), and Exponential(scale=θ).

Usage

```
estimate.uniform(x)
estimate.normal(x)

estimate.gamma(x, use.rate = FALSE)

estimate.logistic(x, eps = 1e-07, verbose = FALSE)

estimate.laplace(x, use.sd = FALSE)

estimate.weibull(x, eps = 1e-07)

estimate.exp(x, use.rate = FALSE)
```

Arguments

`x`	A random sample.
`use.rate`	Logical; if TRUE use the rate instead of the scale.
`eps`	Stopping criterion, 1e-7 by default.
`verbose`	Logical; if TRUE, print estimates in each iteration.
`use.sd`	Logical; if TRUE use the sd instead of the scale for Laplace distribution.

Value

Estimated parameters of the assumed distribution by MLE.

Examples

```
x0=runif(n=100,min=-1,max=1)
estimate.uniform(x0)

x1=rnorm(n=100,mean=0,sd=1)
estimate.normal(x1)

x2=rgamma(n=100,shape=1,scale=1)
estimate.gamma(x2)

x3=rlogis(n=100,location=0,scale=1)
estimate.logistic(x3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
estimate.laplace(x4)

x5=rweibull(n=100,shape=1,scale=1)
estimate.weibull(x5)

x6=rexp(n=100,rate=1/2)
estimate.exp(x6,use.rate=TRUE)
```

gof *Generic GOF tests based on EDF*

Description

This function takes in an i.i.d. random sample, use MLE to estimate parameters of the assumed distribution, compute probability integral transforms, and computes Cramér-von Mises, Anderson-Darling and Watson statistics and their P-values using `imhof` function in `CompQuadForm`.

Usage

```
gof.uniform(x, print = FALSE, verbose = FALSE)

gof.normal(x, print = FALSE, verbose = FALSE)

gof.gamma(x, print = FALSE, verbose = FALSE)

gof.logistic(x, print = FALSE, verbose = FALSE)

gof.laplace(x, print = FALSE, verbose = FALSE)

gof.weibull(x, print = FALSE, verbose = FALSE)

gof.extremevalue(x, print = FALSE, verbose = FALSE)

gof.exp(x, print = FALSE, verbose = FALSE)
```

Arguments

`x`	A random sample.
`print`	Logical; if TRUE print both statistics and P-values; if FALSE the results are returned invisibly.
`verbose`	verbose Logical; if TRUE, print warning messages.

Details

Parameters must be estimated by maximum likelihood (ML) in order for the P-values computed here to be asymptotically valid. They are computed using the fact that when parameters are estimated by maximum likelihood and the null hypothesis is true, the asymptotic distribution of the GOF statistic is the distribution of an infinite weighted sum of weighted chi-square random variables on 1 degree of freedom. The weights are eigenvalues of an integral equation. They depend on the distribution being tested, the statistic being used, and in some cases on the actual parameter values. These weights are then computed approximately by discretization of the integral equation; when that equation depends on one or more parameter values we use the MLE in the equation.

Some notes on the specific distributions: For the Normal, Logistic, Laplace, Extreme Value, Weibull and Exponential distributions, the limiting distributions do not depend on the parameters. For the

Gamma distribution, the shape parameter affects the limiting distribution. The tests remain asymptotically valid when the MLE is used to approximate the limit distribution.

The Exponential distribution is a special case of the Weibull and Gamma families arising when the shape is known to be 1. Knowing a parameter and therefore not estimating it affects the distribution of the test statistic and the functions provided for the Exponential distribution allow for this.

If a data set X_1,...,X_n follows the Weibull distribution then Y_1 = log(X_1), ... ,Y_n = log(X_n) follows the Extreme Value distribution and vice versa. The two procedures give identical test statistics and P-values, in principal.

Some of the models have more than one common parametrization. For the Exponential, Gamma, and Weibull distributions, some writers use a rate parameter and some use the scale parameter which is the inverse of the rate. Our code uses the scale parameter.

For the Laplace distribution, some writers use the density $f(x) = exp(-|x-\mu|/\beta)/(2\beta)$ in which β is a scale parameter. Others use the standard deviation $\sigma = \beta/\sqrt{2}$. Our code uses the scale parameter.

For the Uniform distribution, we offer code for the two parameter Uniform distribution on the range θ_1 to θ_2. These are estimated by the sample minimum and sample maximum. The probability integral transforms of the remaining n-2 points are then tested for uniformity on the range 0 to 1. This procedure is justified because the these probability integral transforms have exactly this distribution if the original data had a uniform distribution over any interval.

It is not unusual to test the hypothesis that a sample follows the standard uniform distribution on [0,1]. In this case the parameters *should not* be estimated. Instead use `AD(z)` or `CvM(z)` or `Watson(z)` to compute the statistic values and then get P-values from `AD.uniform.pvalue(a)` or `CvM.uniform.pvalue(w)` or `Watson.uniform.pvalue(u)` whichever is wanted.

Value

Cramér-von Mises, Anderson-Darling and Watson statistics and their P-values.

See Also

`gof.sandwich` for general distributions using Sandwich estimation of covariance function; `gof.bootstrap` for generic functions using bootstrap method.

Examples

```
x0=runif(n=100,min=-1,max=1)
gof.uniform(x0,print=FALSE)

x1=rnorm(n=100,mean=0,sd=1)
gof.normal(x1)

x2=rgamma(n=100,shape=1,scale=1)
gof.gamma(x2)

x3=rlogis(n=100,location=0,scale=1)
gof.logistic(x3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
```

```
gof.laplace(x4)

x5=rweibull(n=100,shape=1,scale=1)
gof.weibull(x5)
x5_log=log(x5)
gof.extremevalue(x5_log)

x6=rexp(n=100,rate=1/2)
gof.exp(x6)
```

---

gof.bootstrap *Generic GOF tests based on EDF using bootstrap*

Description

This function takes in an i.i.d. random sample, use MLE to estimate parameters of the assumed distribution, compute probability integral transforms, and computes Cramér-von Mises, Anderson-Darling and Watson statistics and their P-values using bootstrap method.

Usage

```
gof.uniform.bootstrap(x, M = 10000)

gof.normal.bootstrap(x, M = 10000)

gof.gamma.bootstrap(x, M = 10000)

gof.logistic.bootstrap(x, M = 10000)

gof.laplace.bootstrap(x, M = 10000)

gof.weibull.bootstrap(x, M = 10000)

gof.extremevalue.bootstrap(x, M = 10000)

gof.exp.bootstrap(x, M = 10000)
```

Arguments

`x`	A random sample.
`M`	Number of bootstrap, 10000 by default.

Value

Cramér-von Mises, Anderson-Darling and Watson statistics and their P-values.

See Also

gof.sandwich for general distributions using Sandwich estimation of covariance function; gof for generic functions using imhof function.

Examples

```
x0=runif(n=100,min=-1,max=1)
gof.uniform.bootstrap(x0,M=100)

x1=rnorm(n=100,mean=0,sd=1)
gof.normal.bootstrap(x1,M=100)

x2=rgamma(n=100,shape=1,scale=1)
gof.gamma.bootstrap(x2,M=100)

x3=rlogis(n=100,location=0,scale=1)
gof.logistic.bootstrap(x3,M=100)

x4= rmutil::rlaplace(n=100,m=0,s=1)
gof.laplace.bootstrap(x4,M=100)

x5=rweibull(n=100,shape=1,scale=1)
gof.weibull.bootstrap(x5,M=100)
x5_log=log(x5)
gof.extremevalue.bootstrap(x5_log,M=100)

x6=rexp(n=100,rate=1/2)
gof.exp.bootstrap(x6,M=100)
```

---

gof.sandwich *GOF tests for general distributions using Sandwich estimation of covariance function*

Description

This function tests the hypothesis that data y come from distribution `Fdist` with unknown parameter values theta. Estimates of theta must be provided in `thetahat`.

Usage

```
gof.sandwich(y, x = NULL, Fdist, thetahat, Score, m = max(n, 100), ...)
```

Arguments

`y`	A random sample or the response of regression problem.
`x`	The matrix of covariates.
`Fdist`	User supplied function to compute probability integral transform of y.
`thetahat`	Parameter estimates by MLE.

`Score`	User supplied function to compute 3 components of the score function an n by p matrix with entries partial log f(y_i,θ)/ partial theta_j.
`m`	Eigenvalues are extracted for an m by m grid of the covariance function.
`...`	Other inputs passed to `Fdist` and `Score` when needed.

Details

It uses a large sample approximation to the limit distribution based on the use of the score function components to estimate the Fisher information and the limiting covariance function of the empirical process.

The estimates `thetahat` should be roots of the likelihood equations.

Value

Cramér-von Mises, Anderson-Darling and Watson statistics and their P-values.

See Also

`gof` for generic functions using `imhof` function; `gof.bootstrap` for generic functions using bootstrap method.

Examples

```
sample = rnorm(n=100,mean=0,sd=1)
mle = estimate.normal(sample)
cdf.normal.user = function(x,theta){
 pnorm(x,mean=theta[1],sd=theta[2])
}
score.normal.user = function(x,theta){
 sig=theta[2]
 mu=theta[1]
 s.mean= (x-mu)/sig
 s.sd= s.mean^2/sig-length(x)/sig
 cbind(s.mean/sig,s.sd)
}
output = gof.sandwich(y=sample,Fdist=cdf.normal.user,thetahat=mle,Score=score.normal.user,m=100)
output
```

`Watson` *Watson statistic*

Description

Compute the Watson goodness-of-fit statistic U^2 for an i.i.d. sample, x, to test for the given distribution with parameters unknown. Estimate parameters by ML using `EDFtest` MLE function by default.

Usage

```
Watson.uniform(x, parameter = estimate.uniform(x))

Watson.normal(x, parameter = estimate.normal(x))

Watson.gamma(x, parameter = estimate.gamma(x))

Watson.logistic(x, parameter = estimate.logistic(x))

Watson.laplace(x, parameter = estimate.laplace(x))

Watson.weibull(x, parameter = estimate.weibull(x))

Watson.extremevalue(x)

Watson.exp(x, parameter = estimate.exp(x))

Watson(z)
```

Arguments

`x`	A random sample.
`parameter`	Parameters of the given distribution, MLE by default.
`z`	A standard uniform random sample.

Value

Watson statistic of the given sample.

See Also

`estimate` for estimating distribution parameters by ML; `CvM` for calculating Cramér-von Mises statistic; `AD` for calculating Anderson-Darling statistic; `Watson.pvalue` for calculating P-value of Watson statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
Watson.uniform(x0)

x1=rnorm(n=100,mean=0,sd=1)
Watson.normal(x1)

x2=rgamma(n=100,shape=1,scale=1)
Watson.gamma(x2)

x3=rlogis(n=100,location=0,scale=1)
Watson.logistic(x3)
```

```
x4= rmutil::rlaplace(n=100,m=0,s=1)
Watson.laplace(x4)

x5=rweibull(n=100,shape=1,scale=1)
Watson.weibull(x5)
x5_log=log(x5)
Watson.extremevalue(x5_log)

x6=rexp(n=100,rate=1/2)
Watson.exp(x6)
```

`Watson.pvalue` *P-value of Watson statistic*

Description

Compute the P-value of the given Watson statistic U^2 using `imhof` function in `CompQuadForm`.

Usage

```
Watson.uniform.pvalue(u, neig = 100, verbose = FALSE)

Watson.normal.pvalue(u, neig = 100, verbose = FALSE)

Watson.gamma.pvalue(u, shape, neig = 100, verbose = FALSE)

Watson.logistic.pvalue(u, neig = 100, verbose = FALSE)

Watson.laplace.pvalue(u, neig = 100, verbose = FALSE)

Watson.weibull.pvalue(u, neig = 100, verbose = FALSE)

Watson.extremevalue.pvalue(u, neig = 100, verbose = FALSE)

Watson.exp.pvalue(u, neig = 100, verbose = FALSE)
```

Arguments

`u`	Watson statistic U^2 with a given distribution.
`neig`	Number of eigenvalues used for `imhof()`.
`verbose`	Logical; if TRUE, print warning messages.
`shape`	The shape parameter of Gamma distribution.

Details

Parameters must be estimated by maximum likelihood (ML) in order for the P-values computed here to be asymptotically valid. They are computed using the fact that when parameters are estimated by maximum likelihood and the null hypothesis is true, the asymptotic distribution of the GOF statistic is the distribution of an infinite weighted sum of weighted chi-square random variables on 1 degree of freedom. The weights are eigenvalues of an integral equation. They depend on the distribution being tested, the statistic being used, and in some cases on the actual parameter values. These weights are then computed approximately by discretization of the integral equation; when that equation depends on one or more parameter values we use the MLE in the equation.

Some notes on the specific distributions: For the Normal, Logistic, Laplace, Extreme Value, Weibull and Exponential distributions, the limiting distributions do not depend on the parameters. For the Gamma distribution, the shape parameter affects the limiting distribution. The tests remain asymptotically valid when the MLE is used to approximate the limit distribution.

The Exponential distribution is a special case of the Weibull and Gamma families arising when the shape is known to be 1. Knowing a parameter and therefore not estimating it affects the distribution of the test statistic and the functions provided for the Exponential distribution allow for this.

If a data set X_1,...,X_n follows the Weibull distribution then Y_1 = log(X_1), ... ,Y_n = log(X_n) follows the Extreme Value distribution and vice versa. The two procedures give identical test statistics and P-values, in principal.

Some of the models have more than one common parametrization. For the Exponential, Gamma, and Weibull distributions, some writers use a rate parameter and some use the scale parameter which is the inverse of the rate. Our code uses the scale parameter.

For the Laplace distribution, some writers use the density $f(x) = exp(-|x-\mu|/\beta)/(2\beta)$ in which β is a scale parameter. Others use the standard deviation $\sigma = \beta/\sqrt{2}$. Our code uses the scale parameter.

For the Uniform distribution, we offer code for the two parameter Uniform distribution on the range θ_1 to θ_2. These are estimated by the sample minimum and sample maximum. The probability integral transforms of the remaining n-2 points are then tested for uniformity on the range 0 to 1. This procedure is justified because the these probability integral transforms have exactly this distribution if the original data had a uniform distribution over any interval.

It is not unusual to test the hypothesis that a sample follows the standard uniform distribution on [0,1]. In this case the parameters *should not* be estimated. Instead use `AD(z)` or `CvM(z)` or `Watson(z)` to compute the statistic values and then get P-values from `AD.uniform.pvalue(a)` or `CvM.uniform.pvalue(w)` or `Watson.uniform.pvalue(u)` whichever is wanted.

Value

P-value of the given Watson statistic.

See Also

`Watson` for calculating Watson statistic; `CvM.pvalue` for calculating P-value of Cramér-von Mises statistic; `AD.pvalue` for calculating P-value of Anderson-Darling statistic.

Examples

```
x0=runif(n=100,min=-1,max=1)
usq0 = Watson.uniform(x0)
Watson.uniform.pvalue(usq0)

x1=rnorm(n=100,mean=0,sd=1)
usq1 = Watson.normal(x1)
Watson.normal.pvalue(usq1)

x2=rgamma(n=100,shape=1,scale=1)
usq2 = Watson.gamma(x2)
Watson.gamma.pvalue(usq2,1)

x3=rlogis(n=100,location=0,scale=1)
usq3 = Watson.logistic(x3)
Watson.logistic.pvalue(usq3)

x4= rmutil::rlaplace(n=100,m=0,s=1)
usq4 = Watson.laplace(x4)
Watson.laplace.pvalue(usq4)

x5=rweibull(n=100,shape=1,scale=1)
usq5 = Watson.weibull(x5)
Watson.weibull.pvalue(usq5)
x5_log=log(x5)
Watson.extremevalue.pvalue(Watson.extremevalue(x5_log))

x6=rexp(n=100,rate=1/2)
usq6 = Watson.exp(x6)
Watson.exp.pvalue(usq6)
```

Index