Package: cvmdisc (via r-universe)

September 17, 2024

Type Package

Title Cramer von Mises Tests for Discrete or Grouped Distributions

Version 0.1.0

Depends R (>= 3.5), stats4, CompQuadForm

Description Implements Cramer-von Mises Statistics for testing fit to (1) fully specified discrete distributions as described in Choulakian, Lockhart and Stephens (1994) <doi:10.2307/3315828> (2) discrete distributions with unknown parameters that must be estimated from the sample data, see Spinelli & Stephens (1997) <doi:10.2307/3315735> and Lockhart, Spinelli and Stephens (2007) <doi:10.1002/cjs.5550350111> (3) grouped continuous distributions with Unknown Parameters, see Spinelli (2001) <doi:10.2307/3316040>. Maximum likelihood estimation (MLE) is used to estimate the parameters. The package computes the Cramer-von Mises Statistics, Anderson-Darling Statistics and the Watson-Stephens Statistics and their p-values.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat

NeedsCompilation no

Author Shaun Zheng Sun [aut, cre], Dillon Duncan [aut]

Maintainer Shaun Zheng Sun <Shaun.Sun@ufv.ca>

Repository CRAN

Date/Publication 2019-11-04 15:30:05 UTC

Contents

`cvmdisc` *Fit Grouped Data*

Description

cvmdisc is a package for fitting data to binomial, poisson, and grouped continuous distributions as well as testing their Goodness of Fit using Cramer-von-Mises and Anderson-Darling statistics

Details

The cvmdisc package includes a function that fits and tests grouped/discrete data to multiple distributions, as well as multiple auxillary functions called by the main function.

Distributions

The Binomial, Poisson, Exponential, Gamma, Log-normal, Normal, Uniform and Weibull distributions are currently supported in the cvmdisc package.

See Also

`groupFit`: Fit data and test GoF

`distrFit`: Fit data

`cvmTest`: Compute test statistics

`cvmPval`: Compute p values

`cvmPval` *cvmPval*

Description

Calculate P values of CVM statistics using their asymptotic distributions

Usage

```
cvmPval(statistic, Msig, imhof = TRUE)
```

Arguments

statistic	test statistic
Msig	Matrix used to produce eigenvalues to estimate the asymptotic distributions of the test statistic
imhof	Logical. Set to be FALSE if Imhof's method is NOT used to approximate the null. The package"CompQuadForm" is required if "imhof=TRUE"

Details

cvmPval is used by groupFit to calculate test statistics for fit distributions.

Value

P-value for test statistic

Author(s)

Shaun Zheng Sun and Dillon Duncan

See Also

groupFit: Data fitting function

Examples

```
# A_squared and MsigA derived from
#(Choulakian, Lockhart and Stephens(1994) Example 3, p8)

A_squared <- 1.172932

MsigA <- matrix(c(0.05000, 0.03829, 0.02061, 0.00644,
                  0.03829, 0.30000, 0.16153, 0.05050,
                  0.02061, 0.16153, 0.30000, 0.09379,
                  0.00644, 0.05050, 0.09379, 0.30000),
                nrow = 4, ncol = 4, byrow = TRUE)

(U2Pval1 = cvmPval(A_squared, MsigA))

U_squared <- 0

MsigU <- matrix(c(0.16666667, 0.10540926, 0.0745356, 0.05270463, 0.03333333,
                  0.10540926, 0.16666667, 0.1178511, 0.08333333, 0.05270463,
                  0.07453560, 0.11785113, 0.1666667, 0.11785113, 0.07453560,
                  0.05270463, 0.08333333, 0.1178511, 0.16666667, 0.10540926,
                  0.03333333, 0.05270463, 0.0745356, 0.10540926, 0.16666667),
                nrow = 5, ncol = 5, byrow = TRUE)

(U2Pval2 = cvmPval(U_squared, MsigU, imhof = FALSE))
```

cvmTest *cvmTest*

Description

Calculate test statistics for grouped data's counts and probabilities

Usage

```
cvmTest(counts, p, pave = FALSE)
```

Arguments

counts	vector containing the frequency of the counts in each group
p	vector of probabilities for counts, often called p-hat
pave	Logical. Set to be FALSE if the probabilities in groups are used; set to TRUE if the average of probabilities of groups j and j+1 are used

Details

cvmTest is used by groupFit to calculate test statistics for the fitted distributions.

Value

A list with the components:

Asq	Anderson-Darling test statistic
Wsq	Cramer-Von-Mises test statistic
Usq	Watson's test statistic
Chisq	Pearson's Chi-squared test statistic
Hj	Estimated cumulative probability

Author(s)

Shaun Zheng Sun and Dillon Duncan

See Also

groupFit: Data fitting function

Examples

```
#Choulakian, Lockhart and Stephens (1994)
counts <- c(10, 19, 18, 15, 11, 13, 7, 10, 13, 23, 15, 22)
phat <- rep(1/12, 12)

(stats1 <- cvmTest(counts, phat))

#Choulakian, Lockhart and Stephens (1994)
counts <- c(1 ,4 ,11 ,4 ,0)
phat <- c(0.05, 0.3, 0.3, 0.3, 0.05)

(stat2 <- cvmTest(counts, phat))

#Utilizing Benford's Law

#Setting pave to TRUE

#genomic data, Lesperance et al (2016)
genomic<- c(48, 14, 12, 6, 18, 5, 7, 8, 9)
phat<- log10(1+1/1:9)

(stat3 <- cvmTest(genomic, phat, pave = TRUE))
```

`distrFit` *distrFit*

Description

Finds the Maximum Likelihood Estimates of the parameters in a requested distribution.

Usage

```
distrFit(breaks, counts, distr, initials)
```

Arguments

`breaks`	Vector defining the breaks in each group
`counts`	Vector containing the frequency of counts in each group
`distr`	Character; the name of the distribution users want to fit the data to distrFit supports all of the continuous distributions supported in `groupFit`:
`initials`	Vector of initial values for the maximum likelihood estimates.

Details

distFit uses Maximum Likelihood Estimates to optimize the parameters for a requested distribution.

Value

distFit returns a vector containing the MLEs.

Author(s)

Shaun Zheng Sun and Dillon Duncan

See Also

groupFit for fitting data and providing GoF statistics.

Examples

```
#fitting exponential data without initial values (Spinelli 2001)

breaks <- c(0, 2, 6, 10, 14, 18, 22, 26)
counts <- c(21, 9, 5, 2, 1, 1, 0)

(mle1 <- distrFit(breaks, counts, distr = "exp"))

#fitting generated data with initial values

breaks <- seq(0, 40, 2)
counts <- table(cut(rweibull(200, 0.5, 3), breaks))

(mle2 <- distrFit(breaks, counts, distr = "weibull", initials = c(0.5, 3)))

#fitting generated data to a different distribution

breaks <- seq(-100, 100, 5)
counts <- table(cut(rcauchy(500, -20, 10), breaks))

(mle3 <- distrFit(breaks, counts, distr = "norm"))
```

---

groupFit *groupFit*

Description

Fits grouped continuous data or discrete data to a distribution and computes Cramer-Von Mises Goodness of Fit statistics and p-values.

Usage

```
groupFit(breaks, counts, data, discrete, distr, N, params, initials,
  pfixed, bootstrap = FALSE, numLoops = 5000, known = FALSE,
  pave = FALSE, imhof = TRUE)
```

Arguments

breaks	Vector defining the breaks in each group.
counts	Vector containing the frequency of counts in each group.
data	Vector containing values of discrete random variables.
discrete	Logical. Is the distribution discrete?
distr	Character; the name of the distribution users want to fit the data to. Included continuous distributions are: "exp", "gamma", "lnorm", "norm", "unif" and "weibull". Included discrete distributions are: "binom" and "pois". User defined distributions are supported as "user". Short-hand or full spelling of these distributions will be recognized, case insensitive.
N	Number of trials, used only for the binomial distribution.
params	Vector of distribution parameters. This is only required when known == TRUE. groupFit will estimate the parameters if known == FALSE.
initials	Vector of distribution parameters to use as starting points for calculating MLEs.
pfixed	Vector of known probabilities for corresponding counts vector. pfixed must be provided when distr = "user".
bootstrap	Logical. Should p-values be calculated via bootstrapping?
numLoops	Number of Bootstrap iterations. Set to be 5000 by default.
known	Logical. Set to be TRUE if the parameters are known and do not need to be estimated.
pave	Logical. Set to be FALSE if the probabilities in groups are used; set to TRUE if the average of probabilities of groups j and j + 1 are used. See page 2 of Spinelli (2001) for more details.
imhof	Logical. Set to be TRUE if Imhof's method from the package "CompQuadForm" is used to approximate the null distributions.

Details

For grouped continuous data: call groupFit with arguments breaks, counts, and distr to fit the data.

For discrete data call groupFit with data and distr to fit the data. If distr = "binom", then be sure to call groupFit with N as well.

Provide initials to suggest starting points for parameter estimation.

Provide params and set known = TRUE to test goodness of fit when parameters are known.

Set bootstrap = TRUE to use bootstrapping to estimate p-values rather than using asymptotic distributions.

Set imhof = FALSE when $a + bX_p^2$ is used to approximate the null distributions. See page 5 of Spinelli (2001) for details.

groupFit can test the fit of user defined discrete distributions. To do so, set distr to "user", and provide the vector pfixed, where each cell contains the probability corresponding to that same cell in counts.

Value

List containing the components:

estimates	The estimated parameters of the distribution
stats	Data frame containing the goodness of fit statistics
pvals	Data frame containing p-values for the goodness of fit statistics

Author(s)

Shaun Zheng Sun and Dillon Duncan

References

V. Choulakian, R. A. Lockhart & M. A. Stephens (1994). Cramer-vonMises statistics for discrete distributions. The Canadian Journal of Statistics,2 2,125-137.

J.J.Spinelli (2001). Testing fit for the grouped exponential distribution. The Canadian Journal of Statistics,29,451-458.

J.J. Spinelli (1994). Cramer-vonMises statistics for discrete distributions. Phd thesis. Simon Fraser University, Burnaby, Canada.

S. Z. Sun, J.J. Spinelli & M. A. Stephens (2012). Testing fit for the grouped gamma and weibull distributions. Technical report. Simon Fraser University, Burnaby, Canada.

See Also

distrFit: Parameter estimation function

Examples

```
#Poisson Example (Spinelli 1994) p36

counts <- c(9, 22, 6, 2, 1, 0, 0, 0, 0)
vals <- 0:8
data <- rep(vals, counts)

groupFit(data = data, distr = "pois")


# When the parameters are unknown
#(Spinelli 1994) p56

counts <- c(57, 203, 383, 525, 532, 408, 273, 139, 45, 27, 10, 4, 0, 1, 1)
vals <- 0:14
data <- rep(vals, counts)

(pois_fit <- groupFit(data = data, distr = "pois"))

#Binomial example when the parameter is unknown
#Spinelli (1994) P92.
N=12
```

```
counts= c(185, 1149, 3265, 5475, 6114, 5194,
          3067, 1331, 403, 105, 14, 4, 0)
vals <- 0:12
data <- rep(vals, counts)

(binom_fit <- groupFit(data = data, N = N, distr = "binom"))

#When the parameter is assumed known and is equal to 1/3

(binom_fit <- groupFit(data = data, N = N, distr = "binom", params = 1/3, known = TRUE))

#uniform example (Choulakian, Lockhart and Stephens(1994) Example 2, p8)

counts <- c(10, 19, 18, 15, 11, 13, 7, 10, 13, 23, 15, 22)

(uni_fit <- groupFit(0:12, counts, distr = "unif"))


#uniform example (Choulakian, Lockhart and Stephens(1994) Example 3, p8)
counts <- c(1, 4, 11, 4, 0)
probability <- c(0.05, 0.3, 0.3, 0.3, 0.05)
breaks <- c(0, cumsum(probability))

#with bootstrapping
(uni_fit1 <- groupFit(breaks, counts, distr = "unif", bootstrap = TRUE, numLoops = 500))

#without bootstrapping
(uni_fit2 <- groupFit(breaks, counts, distr = "unif", bootstrap = FALSE))

#exponential example (Spinelli 2001)

breaks <- c(0, 2, 6, 10, 14, 18, 22, 26)
counts <- c(21, 9, 5, 2, 1, 1, 0)

(exp_fit <- groupFit(breaks, counts, distr = "exp", pave = TRUE))


#Example Sun, Stephens & Spinelli (2012) set 2.
breaks <- c(0, 2, 6, 10, 14, 18, 22, 26)
counts <- c(21, 9, 5, 2, 1, 1, 0)
breaks[1] <- 1e-6
breaks[8] <- Inf

(weibull_fit <- groupFit(breaks, counts, distr = "Weibull"))


#Example Sun, Stephens & Spinelli (2012) set 3.

breaks <- c(0, seq(0.5, 6.5, 1) )
counts <- c(32, 12, 3, 6, 0, 0, 1)
breaks[1] <- 1e-6
breaks[8] <- Inf
```

```
(weibull_fit <- groupFit(breaks, counts, distr = "Weibull"))


#Example Sun, Stephens & Spinelli (2012) set 3.
breaks <- c(0, 2, 6, 10, 14, 18, 22, 26)
counts <- c(21, 9, 5, 2, 1, 1, 0)
breaks[1] <- 1e-6
breaks[8] <- Inf

(gamma_fit <- groupFit(breaks, counts, distr = "exp"))


#Example Sun, Stephens & Spinelli (2012) set 3.
breaks <- c(0, seq(0.5, 6.5, 1) )
counts <- c(32, 12, 3, 6, 0, 0, 1)
breaks[1] <- 1e-6
breaks[8] <- Inf

(gamma_fit <- groupFit(breaks, counts, distr = "gamma"))


#More examples

breaks <- c(0, seq(0.5, 6.5, 1) )
counts <- table(cut(rgamma(100, 3, 1/3), breaks))
breaks[8] <- Inf

#setting pave to true
(exp_fit <- groupFit(breaks, counts, distr = "exp", initials = 0.2, pave = TRUE))

#Setting known to true, with params
(gamma_fit <- groupFit(breaks, counts, distr = "gamma",
                       params = c(3, 1/3), known = TRUE))

#with bootstrapping, specifying the number of loops.
(lnorm_fit <- groupFit(breaks, counts, distr = "lnorm",
                       bootstrap = TRUE, numLoops = 1000))

#fitting with both pave and imhof set to false
#by setting imhof to false, we use a+bX^2_p to approximate
#the distribution of the goodness-of-fit Statistics
(weibull_fit <- groupFit(breaks, counts, distr = "weibull",
                         pave = TRUE, imhof = FALSE))

#Using the user defined distribution to test for Benford's law

#genomic data, Lesperance et al (2016)

genomic <- c(48, 14, 12, 6, 18, 5, 7, 8, 9)
phat <- log10(1+1/1:9)

(fit <- groupFit(counts = genomic, distr = "user", pfixed = phat, imhof = FALSE, pave = TRUE))
```

Index