Package ‘mle.tools’

October 13, 2022

Type Package

Title Expected/Observed Fisher Information and Bias-Corrected Maximum Likelihood Estimate(s)

Version 1.0.0

License GPL (>= 2)

Date 2017-02-21

Author Josmar Mazucheli

Maintainer Josmar Mazucheli <jmazucheli@gmail.com>

Description Calculates the expected/observed Fisher information and the bias-corrected maximum likelihood estimate(s) via Cox-Snell Methodology.

Depends R (>= 3.0.2)

Imports stats

Suggests fitdistrplus (>= 1.0-6)

RoxygenNote 5.0.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-21 15:17:08

R topics documented:

mle.tools-package 2
coxsnell.bc 3
expected.varcov 6
observed.varcov 8

Index **11**

mle.tools-package *Overview of the "mle.tools" Package*

Description

The current version of the **mle.tools** package has implemented three functions which are of great interest in maximum likelihood estimation. These functions calculates the expected /observed Fisher information and the bias-corrected maximum likelihood estimate(s) using the bias formula introduced by Cox and Snell (1968). They can be applied to any probability density function whose terms are available in the derivatives table of `D` function (see "deriv.c" source code for further details). Integrals, when required, are computed numerically via `integrate` function. Below are some mathematical details of how the returned values are calculated.

Let $X_1, \ldots, X_n$ be *i.i.d.* random variables with probability density functions $f(x_i \mid \boldsymbol{\theta})$ depending on a p-dimensional parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$. The *(j,k)-th* element of the observed, H_{jk}, and expected, I_{jk}, Fisher information are calculated, respectively, as

$$H_{jk} = -\sum_{i=1}^{n} \frac{\partial^2}{\partial\theta_j \partial\theta_k} \log f\left(x_i \mid \boldsymbol{\theta}\right)\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}$$

and

$$I_{jk} = -n \times E\left(\frac{\partial^2}{\partial\theta_j \partial\theta_k} \log f\left(x \mid \boldsymbol{\theta}\right)\right) = -n \times \int_{\mathcal{X}} \frac{\partial^2}{\partial\theta_j \partial\theta_k} \log f\left(x \mid \boldsymbol{\theta}\right) \times f\left(x \mid \boldsymbol{\theta}\right) dx\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}$$

where $(j, k = 1, \ldots, p)$, $\widehat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of $\boldsymbol{\theta}$ and $\mathcal{X}$ denotes the support of the random variable X.

The `observed.varcov` function returns the inputted maximum likelihood estimate(s) and the inverse of $\boldsymbol{H}$ while the `expected.varcov` function returns the inputted maximum likelihood estimate(s) and the inverse of $\boldsymbol{I}$. If $\boldsymbol{H}$ and/or $\boldsymbol{I}$ are singular an error message is returned.

Furthermore, the bias corrected maximum likelihood estimate of θ_s $(s = 1, \ldots, p)$, denoted by $\widetilde{\theta}_s$, is calculated as $\widetilde{\theta}_s = \widehat{\theta} - \widehat{Bias}(\widehat{\theta}_s)$, where $\widehat{\theta}_s$ is the maximum likelihood estimate of θ_s and

$$\widehat{Bias}\left(\widehat{\theta}_s\right) = \sum_{j=1}^{p} \sum_{k=1}^{p} \sum_{l=1}^{p} \kappa^{sj} \kappa^{kl} \left[0.5\kappa_{jkl} + \kappa_{jk,l}\right]\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}$$

where κ^{jk} is the *(j,k)-th* element of the inverse of the expected Fisher information, $\kappa_{jkl} = n \times E\left(\frac{\partial^3}{\partial\theta_j \partial\theta_k \theta_l} \log f\left(x \mid \boldsymbol{\theta}\right)\right)$ and $\kappa_{jk,l} = n \times E\left(\frac{\partial^2}{\partial\theta_j \partial\theta_k} \log f\left(x \mid \boldsymbol{\theta}\right) \times \frac{\partial}{\theta_l} \log f\left(x \mid \boldsymbol{\theta}\right)\right)$.

The bias-corrected maximum likelihood estimate(s) and some other quantities are calculated via `coxsnell.bc` function. If the numerical integration fails and/or $\boldsymbol{I}$ is singular an error message is returned.

It is noteworthy that for a series of probability distributions it is possible, after extensive algebra, to obtain the analytical expressions for $Bias(\widehat{\theta}_s)$. In Stosic and Cordeiro (2009) are the analytic expressions for 22 two-parameter continuous probability distributions. They also present the *Maple* and *Mathematica* scripts used to obtain all analytic expressions (see Cordeiro and Cribari-Neto 2014 for further details).

Author(s)

Josmar Mazucheli <jmazucheli@gmail.com>

References

Azzalini, A. (1996). *Statistical Inference: Based on the Likelihood*. London: Chapman and Hall.

Cordeiro, G. M. and Cribari-Neto, F., (2014). An introduction to Bartlett correction and bias reduction. SpringerBriefs in Statistics, New-York.

Cordeiro, G. M. and McCullagh, P., (1991). Bias correction in generalized linear models. *Journal of the Royal Statistical Society, Series B*, **53**, 3, 629–643.

Cox, D. R. and Hinkley, D. V. (1974). *Theoretical Statistics*. London: Chapman and Hall.

Cox, D. R. and Snell, E. J., (1968). A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B*, **30**, 2, 24–275.

Efron, B. and Hinkley, D. V. (1978). Assessing the accuracy of the maximum likelihood estimator: Observed versus expected Fisher information. *Biometrika*, **65**, 3, 457–482.

Pawitan, Y. (2001). *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford: Oxford University Press.

Stosic, B. D. and Cordeiro, G. M., (2009). Using Maple and Mathematica to derive bias corrections for two parameter distributions. *Journal of Statistical Computation and Simulation*, **79**, 6, 751–767.

`coxsnell.bc` *Bias-Corrected Maximum Likelihood Estimate(s)*

Description

`coxsnell.bc` calculates the bias-corrected maximum likelihood estimate(s) using the bias formula introduced by Cox and Snell (1968).

Usage

```
coxsnell.bc(density, logdensity, n, parms, mle, lower = "-Inf",
  upper = "Inf", ...)
```

Arguments

`density`	An expression with the probability density function.
`logdensity`	An expression with the logarithm of the probability density function.
`n`	A numeric scalar with the sample size.
`parms`	A character vector with the parameter name(s) specified in the density and log-density expressions.
`mle`	A numeric vector with the parameter estimate(s).
`lower`	The lower integration limit (lower = “-Inf” is the default).
`upper`	The upper integration limit (upper = “Inf” is the default).
`...`	Additional arguments passed to `integrate` function.

Details

The first, second and third-order partial log-density derivatives are analytically calculated via `D` function. The expected values of the partial log-density derivatives are calculated via `integrate` function.

Value

`coxsnell.bc` returns a list with five components (i) **mle**: the inputted maximum likelihood estimate(s), (ii) **varcov**: the expected variance-covariance evaluated at the inputted mle argument, (iii) **mle.bc**: the bias-corrected maximum likelihood estimate(s), (iv) **varcov.bc**: the expected variance-covariance evaluated at the bias-corrected maximum likelihood estimate(s) and (v) **bias**: the bias estimate(s).

If the numerical integration fails and/or the expected information is singular an error message is returned.

Author(s)

Josmar Mazucheli <jmazucheli@gmail.com>

See Also

deriv, D, expected.varcov, integrate, observed.varcov.

Examples

```
{library(mle.tools); library(fitdistrplus); set.seed(1)};

## Normal distribution
pdf <- quote(1 / (sqrt(2 * pi) * sigma) * exp(-0.5 / sigma ^ 2 * (x - mu) ^ 2))
lpdf <- quote(- log(sigma) - 0.5 / sigma ^ 2 * (x - mu) ^ 2)

x <- rnorm(n = 100, mean = 0.0, sd = 1.0)
{mu.hat <- mean(x); sigma.hat = sqrt((length(x) - 1) * var(x) / length(x))}

coxsnell.bc(density = pdf, logdensity = lpdf, n = length(x), parms = c("mu", "sigma"),
 mle = c(mu.hat, sigma.hat), lower = '-Inf', upper = 'Inf')

################################################################################

## Weibull distribution
pdf <- quote(shape / scale ^ shape * x ^ (shape - 1) * exp(-(x / scale) ^ shape))
lpdf <- quote(log(shape) - shape * log(scale) + shape * log(x) -
 (x / scale) ^ shape)

x <- rweibull(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'weibull')
fit$vcov

coxsnell.bc(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape", "scale"),
 mle = fit$estimate, lower = 0)
```

```
################################################################################

## Exponentiated Weibull distribution
pdf <- quote(alpha * shape / scale ^ shape * x ^ (shape - 1) * exp(-(x / scale) ^ shape) *
 (1 - exp(-(x / scale) ^ shape)) ^ (alpha - 1))
lpdf <- quote(log(alpha) + log(shape) - shape * log(scale) + shape * log(x) -
 (x / scale) ^ shape + (alpha - 1) * log((1 - exp(-(x / scale) ^ shape))))

coxsnell.bc(density = pdf, logdensity = lpdf, n = 100, parms = c("shape", "scale", "alpha"),
 mle = c(1.5, 2.0, 1.0), lower = 0)

################################################################################

## Exponetial distribution
pdf <- quote(rate * exp(-rate * x))
lpdf <- quote(log(rate) - rate * x)

x <- rexp(n = 100, rate = 0.5)

fit <- fitdist(data = x, distr = 'exp')
fit$vcov

coxsnell.bc(density = pdf, logdensity = lpdf, n = length(x), parms = c("rate"),
 mle = fit$estimate, lower = 0)

################################################################################

## Gamma distribution
pdf <- quote(1 /(scale ^ shape * gamma(shape)) * x ^ (shape - 1) * exp(-x / scale))
lpdf <- quote(-shape * log(scale) - lgamma(shape) + shape * log(x) -
 x / scale)

x <- rgamma(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'gamma', start = list(shape = 1.5, scale =  2.0))
fit$vcov

coxsnell.bc(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape", "scale"),
 mle = fit$estimate, lower = 0)

################################################################################

## Beta distribution
pdf <- quote(gamma(shape1 + shape2) / (gamma(shape1) * gamma(shape2)) * x ^ (shape1 - 1) *
 (1 - x) ^ (shape2 - 1))
lpdf <- quote(lgamma(shape1 + shape2) - lgamma(shape1) - lgamma(shape2) +
 shape1 * log(x) + shape2 * log(1 - x))

x <- rbeta(n = 100, shape1 = 2.0, shape2 = 2.0)

fit <- fitdist(data = x, distr = 'beta', start = list(shape1 = 2.0, shape2 =  2.0))
fit$vcov
```

```
coxsnell.bc(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape1", "shape2"),
mle = fit$estimate, lower = 0, upper = 1)
```

`expected.varcov` *Expected Fisher Information*

Description

`expected.varcov` calculates the inverse of the expected Fisher information. Analytical second-order partial log-density derivatives and numerical integration are used in the calculations.

Usage

```
expected.varcov(density, logdensity, n, parms, mle, lower = "-Inf",
  upper = "Inf", ...)
```

Arguments

`density`	An expression with the probability density function.
`logdensity`	An expression with the log of the probability density function.
`n`	A numeric scalar with the sample size.
`parms`	A character vector with the parameter name(s) specified in the density and log-density expressions.
`mle`	A numeric vector with the parameter estimate(s).
`lower`	The lower integration limit (lower = “-Inf” is the default).
`upper`	The upper integration limit (upper = “Inf” is the default).
`...`	Additional arguments passed to `integrate` function.

Details

The second-order partial log-density derivatives and its expected values are calculated via `D` and `integrate` functions, respectively.

Value

`expected.varcov` returns a list with two components (i) **mle**: the inputted maximum likelihood estimate(s) and (ii) **varcov**: the expected variance-covariance evaluated at the inputted mle argument.

If the numerical integration fails and/or the expected information is singular an error message is returned.

Author(s)

Josmar Mazucheli <`jmazucheli@gmail.com`>

See Also

deriv, D, integrate, expected.varcov.

Examples

```
{library(mle.tools); library(fitdistrplus); set.seed(1)};

## Normal distribution
pdf <- quote(1 / (sqrt(2 * pi) * sigma) * exp(-0.5 / sigma ^ 2 * (x - mu) ^ 2))
lpdf <- quote(-log(sigma) - 0.5 / sigma ^ 2 * (x - mu) ^ 2)

x <- rnorm(n = 100, mean = 0.0, sd = 1.0)

expected.varcov(density = pdf, logdensity = lpdf, n = length(x), parms = c("mu", "sigma"),
 mle = c(mean(x), sd(x)), lower = '-Inf', upper = 'Inf')

################################################################################

## Weibull distribution
pdf <- quote(shape / scale ^ shape * x ^ (shape - 1) * exp(-(x / scale) ^ shape))
lpdf <- quote(log(shape) - shape * log(scale) + shape * log(x) -
 (x / scale) ^ shape)

x <- rweibull(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'weibull')
fit$vcov

expected.varcov(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape", "scale"),
 mle = fit$estimate, lower = 0)

################################################################################

## Expoentiated Weibull distribution
pdf <- quote(alpha * shape / scale ^ shape * x ^ (shape - 1) * exp(-(x / scale) ^ shape) *
 (1 - exp(-(x / scale) ^ shape)) ^ (alpha - 1))
lpdf <- quote(log(alpha) + log(shape) - shape * log(scale) + shape * log(x) -
 (x / scale) ^ shape + (alpha - 1) * log((1 - exp(-(x / scale) ^ shape))))

expected.varcov(density = pdf, logdensity = lpdf, n = 100, parms = c("shape", "scale", "alpha"),
 mle = c(1.5, 2.0, 1.0), lower = 0)
################################################################################

## Exponetial distribution
pdf <- quote(rate * exp(-rate * x))
lpdf <- quote(log(rate) - rate * x)

x <- rexp(n = 100, rate = 0.5)

fit <- fitdist(data = x, distr = 'exp')
fit$vcov
```

```
expected.varcov(density = pdf, logdensity = lpdf, n = length(x), parms = c("rate"),
 mle = fit$estimate, lower = 0)

################################################################################

## Gamma distribution
pdf <- quote(1 /(scale ^ shape * gamma(shape)) * x ^ (shape - 1) * exp(-x / scale))
lpdf <- quote(-shape * log(scale) - lgamma(shape) + shape * log(x) -
 x / scale)

x <- rgamma(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'gamma', start = list(shape = 1.5, scale =  2.0))
fit$vcov

expected.varcov(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape", "scale"),
 mle = fit$estimate, lower = 0)

################################################################################

## Beta distribution
pdf <- quote(gamma(shape1 + shape2) / (gamma(shape1) * gamma(shape2)) * x ^ (shape1 - 1) *
(1 - x) ^ (shape2 - 1))
lpdf <- quote(lgamma(shape1 + shape2) - lgamma(shape1) - lgamma(shape2) +
 shape1 * log(x) + shape2 * log(1 - x))

x <- rbeta(n = 100, shape1 = 2.0, shape2 = 2.0)

fit <- fitdist(data = x, distr = 'beta', start = list(shape1 = 2.0, shape2 =  2.0))
fit$vcov

expected.varcov(density = pdf, logdensity = lpdf, n = length(x), parms = c("shape1", "shape2"),
 mle = fit$estimate, lower = 0, upper = 1)
```

`observed.varcov` *Observed Fisher Information*

Description

`observed.varcov` calculates the inverse of the observed Fisher Information. Analytical second-order partial log-density derivatives are used in the calculations.

Usage

```
observed.varcov(logdensity, X, parms, mle)
```

Arguments

`logdensity`	An expression with the log of the probability density function.
`X`	A numeric vector with the observations.
`parms`	A character vector with the parameter name(s) specified in the logdensity expression.
`mle`	A numeric vector with the parameter estimate(s).

Details

The second-order partial log-density derivatives are calculated via `D` function.

Value

`observed.varcov` returns a list with two components (i) **mle**: the inputted maximum likelihood estimate(s) and (ii) **varcov**: the observed variance-covariance evaluated at the inputted mle argument.

If the observed information is singular an error message is returned.

Author(s)

Josmar Mazucheli <`jmazucheli@gmail.com`>

See Also

`deriv`, `D`, `expected.varcov`.

Examples

```
{library(mle.tools); library(fitdistrplus); set.seed(1)};

##Normal distribution
lpdf <- quote(-log(sigma) - 0.5 / sigma ^ 2 * (x - mu) ^ 2)

x <- rnorm(n = 100, mean = 0.0, sd = 1.0)

observed.varcov(logdensity = lpdf, X = x, parms = c("mu", "sigma"),
 mle = c(mean(x), sd(x)))

################################################################################

## Weibull distribution
lpdf <- quote(log(shape) - shape * log(scale) + shape * log(x) - (x / scale) ^ shape)

x <- rweibull(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'weibull')
fit$vcov

observed.varcov(logdensity = lpdf, X = x, parms = c("shape", "scale"), mle = fit$estimate)

################################################################################
```

```
## Exponetial distribution
lpdf <- quote(log(rate) - rate * x)

x <- rexp(n = 100, rate = 0.5)

fit <- fitdist(data = x, distr = 'exp')
fit$vcov

observed.varcov(logdensity = lpdf, X = x, parms = c("rate"), mle = fit$estimate)

################################################################################

## Gamma distribution
lpdf <- quote(-shape * log(scale) - lgamma(shape) + shape * log(x) -
 x / scale)

x <- rgamma(n = 100, shape = 1.5, scale = 2.0)

fit <- fitdist(data = x, distr = 'gamma', start = list(shape = 1.5, scale =  2.0))
fit$vcov

observed.varcov(logdensity = lpdf, X = x, parms = c("shape", "scale"), mle = fit$estimate)

################################################################################

## Beta distribution
lpdf <- quote(lgamma(shape1 + shape2) - lgamma(shape1) - lgamma(shape2) +
  shape1 * log(x) + shape2 * log(1 - x))

x <- rbeta(n = 100, shape1 = 2.0, shape2 = 2.0)

fit <- fitdist(data = x, distr = 'beta', start = list(shape1 = 2.0, shape2 =  2.0))
fit$vcov

observed.varcov(logdensity = lpdf, X = x, parms = c("shape1", "shape2"), mle = fit$estimate)
```

Index

coxsnell.bc, 3

D, *4*, *7*, *9*
deriv, *4*, *7*, *9*

expected.varcov, *4*, 6, *7*, *9*

integrate, *4*, *7*

mle.tools-package, 2

observed.varcov, *4*, 8