Package 'ihs'

February 25, 2015

Version 1.0

Type Package

Title Inverse Hyperbolic Sine Distribution

Date 2015-02-24

Author Carter Davis

Maintainer Carter Davis <carterdavis@byu.edu>

Depends R (>= 2.4.0), maxLik (>= 1.2-4)

Description Density, distribution function, quantile function and random generation for the inverse hyperbolic sine distribution. This package also provides a function that can fit data to the inverse hyperbolic sine distribution using maximum likelihood estimation.

License GPL (>= 3)

Repository CRAN

NeedsCompilation no

Date/Publication 2015-02-25 08:12:31

R topics documented:

`ihs` *The Inverse Hyperbolic Sine Distribution*

Description

Density, distribution function, quantile function and random generation for the inverse hyperbolic sine distribution.

Usage

```
dihs(x, mu = 0, sigma = SIGCONST, lambda = 0, k = 1, log = FALSE)
pihs(q, mu = 0, sigma = SIGCONST, lambda = 0, k = 1, lower.tail = TRUE,
log.p = FALSE)
qihs(p, mu = 0, sigma = SIGCONST, lambda = 0, k = 1, lower.tail = TRUE,
log.p = FALSE)
rihs(n, mu = 0, sigma = SIGCONST, lambda = 0, k = 1)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
mu	vector of means. The default value is 0.
sigma	vector of standard deviations. The default value is sqrt((exp(2)-1)/2).
lambda	vector of skewness parameters. If lambda < 0, the distribution is skewed to the left. If lambda > 0, the distribution is skewed to the right. If lambda = 0, then the distribution is symmetric.
k	vector of parameters. This parameter controls the skewness of the distribution.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.

Details

If mu, sigma, lambda, or k are not specified they assume the default values of mu = 0, sigma is approximately sqrt((exp(2)-1)/2), lambda = 0, and k = 1. These default values give the distribution sinh(z), where z is a standard normal random variable.

An inverse hyperbolic sine random variable Y is defined by the transformation

$$Y = a + b * sinh(\lambda + Z/k)$$

where Z is a standard normal random variable, and a, b, λ, and k control the mean, variance, skewness, and kurtosis respectively. Thus the inverse hyperbolic sine distribution has density

$$f(x) = \frac{ke^{(-k^2/2)(log(\frac{x-a}{b}+(\frac{(x-a)^2}{b^2}+1)^{1/2})-\lambda)^2}}{\sqrt{2\pi((a-x)^2+b^2)}}$$

and if we reparametrize the distribution so that the parameters include the mean (μ) and the standard deviation (σ) instead of a and b, then we let

$$b = \frac{2\sigma}{\sqrt{(e^{2\lambda+k^{-2}} + e^{-2\lambda+k^{-2}} + 2)(e^{k^{-2}} - 1)}}$$

$$a = \mu - \frac{b}{2}((e^{\lambda} - e^{-\lambda})e^{\frac{1}{2k^2}})$$

Thus if $\mu = 0$, $\sigma = \sqrt{\frac{e^2-1}{2}}$, $\lambda = 0$, and $k = 1$, then $Y = sinh(Z)$.

Value

dihs gives the density, pihs gives the distribution function, qihs gives the quantile function, and rihs generates random deviates.

The length of the result is determined by n for rihs, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

sigma <= 0 and k <= 0 are errors and return NaN.

Author(s)

Carter Davis, <carterdavis@byu.edu>

Source

dihs is calculated from the definition (in 'Details'). [pqr]ihs are based on the relationship to the normal.

References

Hansen, C., McDonald, J. B., and Theodossiou, P. (2007) "Some Flexible Parametric Models for Partially Adaptive Estimators of Econometric Models" *Economics - The Open-Access, Open-Assessment E-Journal*, volume 1, 1-20.

Hansen, C., McDonald, J. B., and Newey, W. K. (2010) "Instrumental Variables Regression with Flexible Distributions" *Journal of Business and Economic Statistics*, volume 28, 13-25.

See Also

Distributions for other standard distributions such as dnorm for the normal distribution and dlnorm for the log-normal distribution, which is also a transformation of a normal random variable.

Examples

```
require(graphics)

### This shows how default values of the IHS compare
### to a standard normal.
x = seq(-5,5,by=0.05)
plot(x, dnorm(x), type='l')
lines(x, dihs(x), col='blue')

pihs(0)
pihs(0, lambda = -0.5)
```

ihsmle *Maximum Likelihood Estimation with the Inverse Hyperbolic Sine Distribution*

Description

This function allows data to be fit to the inverse hyperbolic sine distribution using maximum likelihood estimation. This function uses the `maxLik` package to perform its estimations.

Usage

```
ihs.mle(X.f, mu.f = mu ~ mu, sigma.f = sigma ~ sigma,
lambda.f = lambda ~ lambda, k.f = k ~ k, data = parent.frame(),
start, subset, method = 'BFGS', constraints = NULL,
follow.on = FALSE, iterlim = 5000, ...)
```

Arguments

`X.f`	A formula specifying the data, or the function of the data with parameters, that should be used in the maximisation procedure. `X` should be on the left-hand side and the right-hand side should be the data or function of the data that should be used.
`mu.f, sigma.f, lambda.f, k.f`	formulas including variables and parameters that specify the functional form of the parameters in the inverse hyperbolic sine log-likelihood function. `mu`, `sigma`, `lambda`, and `k` should be on the left-hand side of these formulas respectively.
`data`	an optional data frame in which to evaluate the variables in `formula` and `weights`. Can also be a list or an environment.
`start`	a named list or named numeric vector of starting estimates for every parameter.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`method`	A list of the methods to be used. May include "NR" (for Newton-Raphson), "BFGS" (for Broyden-Fletcher-Goldfarb-Shanno), "BHHH" (for Berndt-Hall-Hall-Hausman), "SANN" (for Simulated ANNealing), "CG" (for Conjugate Gradients), or "NM" (for Nelder-Mead). Lower-case letters (such as "nr" for Newton-Raphson) are allowed. The default method is the "BFGS" method.
`constraints`	either `NULL` for unconstrained optimization or a list with two components. The components may be either `eqA` and `eqB` for equality-constrained optimization $A\theta + B = 0$; or `ineqA` and `ineqB` for inequality constraints $A\theta + B > 0$. More than one row in `ineqA` and `ineqB` corresponds to more than one linear constraint, in that case all these must be zero (equality) or positive (inequality constraints).
`follow.on`	logical; if TRUE, and there are multiple methods, then the last set of parameters from one method is used as the starting set for the next.
`iterlim`	If provided as a vector of the same length as `method`, gives the maximum number of iterations or function values for the corresponding method. If a single number is provided, this will be used for all methods.

... further arguments that are passed to the selected maximisation routine in the maxLik package. See below for a non-exhaustive list of some further arguments that can be used.

Details

The parameter names are taken from start. If there is a name of a parameter or some data found on the right-hand side of one of the formulas but not found in data and not found in start, then an error is given.

Below is a non-exhaustive list of further arguments that may be passed in to the ihs.mle function (see maxLik documentation for more details):

fixed parameters that should be fixed at their starting values: a vector of character strings indicating the names of the fixed parameters (parameter names are taken from argument start). May not be used in BHHH algorithm.

print.level a larger number prints more working information.

tol, reltol the absolute and relative convergence tolerance (see optim). May not be used in BHHH algorithm.

finalHessian how (and if) to calculate the final Hessian. Either FALSE (not calculate), TRUE (use analytic/numeric Hessian) or "bhhh"/"BHHH" for information equality approach.

parscale A vector of scaling values for the parameters. Optimization is performed on 'par/parscale' and these should be comparable in the sense that a unit change in any element produces about a unit change in the scaled value. (see optim). May not be used in BHHH algorithm.

Note that not all arguments may be used for every maximisation algorithm at this time. If multiple methods are supplied (i.e. length(method) > 1), all arguments are employed for each method (except for iterlim, which is allowed to vary for different methods).

If multiple methods are supplied, and some methods fail to initialise properly, a warning will be given. If every method fails to initialise, an error is given.

Value

If only one method is specified, ihs.mle returns a list of class "MLE". If multiple methods are given, ihs.mle returns a list of class "mult.MLE" with each component containing the results of each maximisation procedure. Each component is a list of class "MLE". A list of class "MLE" has the following components:

parameters	the names of the given parameters taken from start
maximum	fn value at maximum (the last calculated value if not converged).
estimate	estimated parameter value.
gradient	vector, last gradient value which was calculated. Should be close to 0 if normal convergence.
gradientObs	matrix of gradients at parameter value estimate evaluated at each observation (only if grad returns a matrix or grad is not specified and fn returns a vector).
hessian	Hessian at the maximum (the last calculated value if not converged).
code	return code:

- 1 gradient close to zero (normal convergence).
- 2 successive function values within tolerance limit (normal convergence).
- 3 last step could not find higher value (probably not converged). This is related to line search step getting too small, usually because hitting the boundary of the parameter space. It may also be related to attempts to move to a wrong direction because of numerical errors. In some cases it can be helped by changing `steptol`.
- 4 iteration limit exceeded.
- 5 Infinite value.
- 6 Infinite gradient.
- 7 Infinite Hessian.
- 8 Successive function values withing relative tolerance limit (normal convergence).
- 9 (BFGS) Hessian approximation cannot be improved because of gradient did not change. May be related to numerical approximation problems or wrong analytic gradient.
- 100 Initial value out of range.

`message` a short message, describing `code`.

`last.step` list describing the last unsuccessful step if `code=3` with following components:

- theta0 previous parameter value
- f0 `fn` value at `theta0`
- climb the movement vector to the maximum of the quadratic approximation

`fixed` logical vector, which parameters are constants.

`iterations` number of iterations.

`type` character string, type of maximization.

`constraints` A list, describing the constrained optimization (`NULL` if unconstrained). Includes the following components:

- type type of constrained optimization
- outer.iterations number of iterations in the constraints step
- barrier.value value of the barrier function

Author(s)

Carter Davis, <carterdavis@byu.edu>

References

Henningsen, Arne and Toomet, Ott (2011). "maxLik: A package for maximum likelihood estimation in R" *Computational Statistics* 26(3), 443-458. DOI 10.1007/s00180-010-0217-1.

See Also

The `maxLik` package and its documentation. The `ihs.mle` simply uses its functions to maximize the inverse hyperbolic sine log-likelihood.

Examples

```
### Showing how to fit a simple vector of data to the inverse
### hyperbolic sine distribution.
require(graphics)
require(stats)
set.seed(123456)
x = rnorm(100)
X.f = X ~ x
start = list(mu = 0, sigma = 2, lambda = 0, k = 1)
result = ihs.mle(X.f = X.f, start = start)
sumResult = summary(result)
print(result)
coef(result)
print(sumResult)

### Comparing the fit
xvals = seq(-5, 5, by = 0.05)
coefs = coef(result)
mu = coefs[1]
sigma = coefs[2]
lambda = coefs[3]
k = coefs[4]
plot(xvals, dnorm(xvals), type = "l", col = "blue")
lines(xvals, dihs(xvals, mu = mu, sigma = sigma,
lambda = lambda, k = k), col = "red")
```

`summary.MLE` *Summary the Maximum-Likelihood Estimation with the Inverse Hyperbolic Sine Distribution*

Description

Summary the maximum-likelihood estimation including standard errors and t-values.

Usage

```
## S3 method for class 'MLE'
summary(object, ...)
## S3 method for class 'mult.MLE'
summary(object, ...)
```

Arguments

`object`	object of class `'MLE'` or of class `'mult.MLE'`, usually a result from maximum-likelihood estimation.
`...`	currently not used.

Value

summary.MLE returns an object of class 'summary.MLE' with the following components:

parameters	names of parameters used in the estimation procedure.
type	type of maximisation.
iterations	number of iterations.
code	code of success.
message	a short message describing the code.
loglik	the loglik value in the maximum.
estimate	numeric matrix, the first column contains the parameter estimates, the second the standard errors, third t-values and fourth corresponding probabilities.
fixed	logical vector, which parameters are treated as constants.
NActivePar	number of free parameters.
constraints	information about the constrained optimization. Passed directly further from maxim-object. NULL if unconstrained maximization.

summary.mult.MLE returns a list of class 'summary.mult.MLE' with components of class 'summary.MLE'.

Author(s)

Carter Davis, <carterdavis@byu.edu>

See Also

the maxLik CRAN package

Examples

```
### Showing how to fit a simple vector of data to the inverse
### hyperbolic sine distribution.
require(graphics)
require(stats)
set.seed(123456)
x = rnorm(100)
X.f = X ~ x
start = list(mu = 0, sigma = 2, lambda = 0, k = 1)
result = ihs.mle(X.f = X.f, start = start)
sumResult = summary(result)
print(result)
coef(result)
print(sumResult)

### Comparing the fit
xvals = seq(-5, 5, by = 0.05)
coefs = coef(result)
mu = coefs[1]
sigma = coefs[2]
lambda = coefs[3]
```

```
k = coefs[4]
plot(xvals, dnorm(xvals), type = "l", col = "blue")
lines(xvals, dihs(xvals, mu = mu, sigma = sigma,
lambda = lambda, k = k), col = "red")
```

Index