Package 'stat.extend'

October 8, 2020

Type Package

Title Highest Density Regions and Other Functions of Distributions

Version 0.1.4

Copyright Ben O'Neill 2020

Description Highest Density Regions are the smallest set in the support of a probability distribution with the specified coverage probability. 'HDRs' may contain disjoint intervals, but can be calculated efficiently using iterative methods. One can similarly construct optimal (i.e., shortest) confidence intervals for some basic inferential problems, including for population means, variances, or proportion parameters.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports sets

Suggests extraDistr, invgamma, VGAM

RoxygenNote 7.0.2

NeedsCompilation no

Author Ben O'Neill [aut, cph],
Neal Fultz [cre, ctb]

Maintainer Neal Fultz <nfultz@gmail.com>

Repository CRAN

Date/Publication 2020-10-08 16:10:02 UTC

R topics documented:

CONF 2
HDR 4
HDR.discrete 12
HDR.monotone 13
reformat 15

Index **16**

CONF *Optimal Confidence Intervals for finite populations*

Description

These functions compute an optimised confidence interval for statistics based on a sample. The user may enter either a data vector `x` or the sample size `n` and the sample statistic. By default the confidence interval is computed for an infinite population. However, the user may enter a population size `N` and may use the logical value `unsampled` to specify when the confidence interval is for the variance only of the unsampled part of the population. This test accounts for the kurtosis, and so the user must either specify the data vector or specify an assumed kurtosis `kurt`; if no kurtosis value is specified then the test uses the sample kurtosis from the data.

Usage

```
CONF.mean(
  alpha,
  x = NULL,
  sample.mean = mean(x),
  sample.variance = var(x),
  n = length(x),
  N = Inf,
  kurt = 3,
  unsampled = FALSE,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

CONF.var(
  alpha,
  x = NULL,
  sample.variance = var(x),
  n = length(x),
  N = Inf,
  kurt = NULL,
  unsampled = FALSE,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

CONF.prop(
  alpha,
  x = NULL,
  sample.prop = mean(x),
  n = length(x),
```

```
  N = Inf,
  unsampled = FALSE
)
```

Arguments

alpha	alpha Numeric (probability) The significance level determining the confidence level for the interval (the confidence level is 1-alpha).
x	Numeric (vector) The vector of sample data. In the CONF.prop function this must be binary data. Ignored if a sample statistic is provided.
sample.mean	Numeric (any) The sample mean of the data.
sample.variance	Numeric (non-neg) The sample variance of the data.
n	Integer (positive) The sample size
N	Integer (positive) The population size (must be at least as large as the sample size)
kurt	Numeric (positive) The assumed kurtosis of the underlying distribution (must be at least one)
unsampled	Logical (positive) Indicator of whether the user wants a confidence interval for the relevant parameter only for the unsampled part of the population (as opposed to the whole population)
gradtol	Parameter for the nlm optimisation - a positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate the algorithm (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html))
steptol	Parameter for the nlm optimisation - a positive scalar providing the minimum allowable relative step length (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
iterlim	Parameter for the nlm optimisation - a positive integer specifying the maximum number of iterations to be performed before the program is terminated (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
sample.prop	Numeric (probability) The sample proportion of the data (only for binary data)

Details

The mean interval is built on a symmetric pivotal quantity so it is symmetric around the sample mean.

The variance interval is built on a non-symmetric pivotal quantity, so it is optimised by taking the shortest possible confidence interval with the specified confidence level (see e.g., Tate and Klett 1959).

The proportion interval uses the Wilson score interval (see e.g., Agresti and Coull 1998).

Value

an object of class 'ci' providing the confidence interval and related information.

Examples

```
DATA <- c(17.772, 16.359, 15.734, 15.698, 16.042,
15.527, 16.533, 15.385, 15.368, 18.603,
15.036, 13.873, 14.329, 15.837, 14.189,
15.398, 16.266, 12.970, 15.219, 16.444,
11.049, 14.262);
KURT <- 4.37559247659433 # moments::kurtosis(DATA);
CONF.mean(alpha = 0.1, x = DATA, N = 3200, kurt = KURT);
CONF.var(alpha = 0.1, x = DATA, N = 3200, kurt = KURT);
CONF.prop(alpha = 0.1, x = DATA > 15, N = 3200);
```

HDR *Highest density region (HDR)*

Description

HDR.xxxx returns the highest density region (HDR) for a chosen distribution.

Usage

```
HDR.norm(
  cover.prob,
  mean = 0,
  sd = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.lnorm(
  cover.prob,
  meanlog = 0,
  sdlog = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.t(cover.prob, df, ncp = 0, gradtol = 1e-10, steptol = 1e-10, iterlim = 100)

HDR.cauchy(
  cover.prob,
  location = 0,
  scale = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
```

```
  iterlim = 100
)

HDR.f(
  cover.prob,
  df1,
  df2,
  ncp = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.beta(
  cover.prob,
  shape1,
  shape2,
  ncp = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.chisq(
  cover.prob,
  df,
  ncp = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.gamma(
  cover.prob,
  shape,
  rate = 1,
  scale = 1/rate,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.weibull(
  cover.prob,
  shape,
  scale = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
```

```
  iterlim = 100
)

HDR.exp(cover.prob, rate, gradtol = 1e-10, steptol = 1e-10, iterlim = 100)

HDR.unif(
  cover.prob,
  min = 0,
  max = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.hyper(cover.prob, m, n, k, gradtol = 1e-10, steptol = 1e-10, iterlim = 100)

HDR.geom(cover.prob, prob, gradtol = 1e-10, steptol = 1e-10, iterlim = 100)

HDR.binom(
  cover.prob,
  size,
  prob,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.pois(cover.prob, lambda, gradtol = 1e-10, steptol = 1e-10, iterlim = 100)

HDR.nbinom(
  cover.prob,
  size,
  prob,
  mu,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.arcsine(
  cover.prob,
  min = 0,
  max = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)
```

```
HDR.betapr(
  cover.prob,
  shape1,
  shape2,
  scale = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.fatigue(
  cover.prob,
  alpha,
  beta = 1,
  mu = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.gompertz(
  cover.prob,
  shape = 1,
  scale = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.gpd(
  cover.prob,
  mu = 0,
  sigma = 1,
  xi = 0,
  location = mu,
  scale = sigma,
  shape = xi,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.huber(
  cover.prob,
  mu,
  sigma,
  epsilon,
  gradtol = 1e-10,
```

```
  steptol = 1e-10,
  iterlim = 100
)

HDR.kumar(
  cover.prob,
  a = 1,
  b = 1,
  shape1 = a,
  shape2 = b,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.tnorm(
  cover.prob,
  mean = 0,
  sd = 1,
  a = -Inf,
  b = Inf,
  min = a,
  max = b,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.invchisq(
  cover.prob,
  df,
  ncp = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.invexp(
  cover.prob,
  rate = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.invgamma(
  cover.prob,
  shape,
```

```
  rate = 1,
  scale = 1/rate,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.benini(
  cover.prob,
  shape,
  y0,
  scale = y0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.frechet(
  cover.prob,
  shape,
  scale = 1,
  location = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.gengamma(
  cover.prob,
  d,
  k,
  shape1 = d,
  shape2 = k,
  rate = 1,
  scale = 1/rate,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.gumbelII(
  cover.prob,
  shape,
  scale = 1,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)
```

```
HDR.lgamma(
  cover.prob,
  shape = 1,
  scale = 1,
  location = 0,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)
```

Arguments

cover.prob	The probability coverage for the HDR (scalar between zero and one). The significance level for the HDR i is 1-cover.prob.
gradtol	Parameter for the nlm optimisation - a positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate the algorithm (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html))
steptol	Parameter for the nlm optimisation - a positive scalar providing the minimum allowable relative step length (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
iterlim	Parameter for the nlm optimisation - a positive integer specifying the maximum number of iterations to be performed before the program is terminated (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
shape1, shape2, ncp, location, scale, df, rate, df1, df2, meanlog, sdlog, mean, sd, min, max, shape, size, p	Distribution parameters.

Details

This function computes the highest density region (HDR) for a univariate distribution in the stats package. The functions for the HDR for different distributions are named in the form HDR.xxxx where the xxxx refers to the distribution (e.g., HDR.chisq, HDR.gamma, HDR.norm, etc.). The user can use any univariate distribution in the stats package, and the function accepts parameters from the specified distribution (see table below). The output of the function is an interval of classes hdr and interval giving the highest density region and some related information pertaining to the distribution and the computation of the HDR (for information on intervals, see the sets package). If the input distribution is continuous then the HDR is a real interval, and if the input distribution discrete then the HDR is a discrete interval. For non-trivial cases the computation is done by optimisation using the nlm function.

Using stats	Continuous		
HDR.arcsine	min	max	
HDR.beta	shape1	shape2	ncp
HDR.cauchy	location	scale	
HDR.chisq	df	ncp	
HDR.exp	rate		
HDR.f	df1	df2	ncp
HDR.gamma	shape	rate	scale

HDR.lnorm	meanlog	sdlog	
HDR.norm	mean	sd	
HDR.t	df	ncp	
HDR.unif	min	max	
HDR.weibull	shape	scale	
Using `stats`	Discrete		
HDR.binom	size	prob	
HDR.geom	prob		
HDR.hyper	m	n	k
HDR.nbinom	size	prob	mu
HDR.pois	lambda		
Using `extraDistr`			
HDR.betapr	shape1	shape2	scale
HDR.fatigue	alpha	beta	mu
HDR.gompertz	shape	scale	
HDR.gpd	mu,location	sigma, scale	shape, xi
HDR.huber	mu	sigma	epsilon
HDR.kumar	a,shape1	b,shape2	
HDR.tnorm	mean	sd	a, b, min, max
Using `invgamma`			
HDR.invchisq	df	ncp	
HDR.invexp	rate		
HDR.invgamma	shape	rate	scale
Using `VGAM`			
HDR.benini	shape	y0	scale
HDR.frechet	shape	scale	location
HDR.gengamma	d, shape1	k, shape2	rate, scale
HDR.gumbelII	shape	scale	
HDR.lgamma	shape	scale	location

The table above shows the parameters in each of the distributions. Some have default values, but most need to be specified. (For the gamma distribution you should specify either the `rate` or `scale` but not both.)

Value

An interval object with classes `hdr` and `interval` containing the highest density region and related information.

Examples

```
HDR.norm(.95)
```

HDR.discrete *Highest density region (HDR) for an arbitrary discrete distribution*

Description

This function computes the highest density region (HDR) with support on the integers. The distribution can be any discrete distribution concentrated on the integers — it does not have to have any shape properties for the function to work. The user must give the density function "'f"' for the distribution. To improve the search properties of the algorithm, the user can also give lower and upper bounds for the support of the distribution if these are available. (Warning: If the user specifies incorrect bounds on the support, that do not contain the full support of the distribution, then the algorithm may continue to search without end, in which case the function will not terminate. Similarly, if the user specifies a sequence function E that is not a proper bijection to the integers then the algorithm may continue to search without end, in which case the function will not terminate.) The output of the function is a `'hdr'` object containing the HDR for the discrete distribution.

Usage

```
HDR.discrete(
  cover.prob,
  f,
  supp.min = -Inf,
  supp.max = Inf,
  E = NULL,
  ...,
  distribution = "an unspecified input distribution"
)
```

Arguments

`cover.prob`	The minimum coverage probability for the region
`f`	The density (mass) function for the distribution
`supp.min`	A minimum bound for the support of the distribution
`supp.max`	A maximum bound for the support of the distribution
`E`	A bijective function mapping the natural numbers (1,2,3,...) to a set covering the support of the distribution (optional); if included, the algorithm will search the support of the distribution in the order specified by this function; if not included, the algorithm will search the integers in a default order.
`...`	additional parameters of f
`distribution`	a label

Value

If all inputs are correctly specified (i.e., arguments and parameters are in allowable range) then the output will be a list of class "'hdr"' containing the HDR and related information.

HDR.monotone *Highest density region (HDR) for an arbitrary distributions*

Description

Highest density region (HDR) for an arbitrary distributions

Usage

```
HDR.monotone(
  cover.prob,
  Q,
  decreasing = TRUE,
  distribution = UNSPECIFIED_LABEL,
  ...
)

HDR.unimodal(
  cover.prob,
  Q,
  f = NULL,
  u = NULL,
  distribution = UNSPECIFIED_LABEL,
  ...,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.bimodal(
  cover.prob,
  Q,
  f = NULL,
  u = NULL,
  distribution = UNSPECIFIED_LABEL,
  ...,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)

HDR.discrete.unimodal(
  cover.prob,
  Q,
  F,
  f = NULL,
  u = NULL,
```

```
  distribution = UNSPECIFIED_LABEL,
  ...,
  gradtol = 1e-10,
  steptol = 1e-10,
  iterlim = 100
)
```

Arguments

cover.prob	The probability coverage for the HDR (scalar between zero and one). The significance level for the HDR i is 1-cover.prob.
Q	an inverse CDF of a distribution
decreasing	Direction of monotone distribution
distribution	a label
...	Arguments for Q, f and u
f	a PDF of a distribution
u	a log-derivative of f
gradtol	Parameter for the nlm optimisation - a positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate the algorithm (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html))
steptol	Parameter for the nlm optimisation - a positive scalar providing the minimum allowable relative step length (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
iterlim	Parameter for the nlm optimisation - a positive integer specifying the maximum number of iterations to be performed before the program is terminated (see [nlm doccumentation](https://stat.ethz.ch/R-manual/R-patched/library/stats/html/nlm.html)).
F	a CDF of a distribution

Value

An interval object with classes hdr and interval containing the highest density region and related information.

See Also

HDR.discrete

Examples

```
HDR.monotone(.95, Q=qexp)

HDR.unimodal(.95, Q=qnorm)

HDR.bimodal(.95, Q=qbeta, shape1=1/2, shape2=1/2)

HDR.discrete.unimodal(.95, Q=qpois, F=ppois, lambda=1)
```

`reformat` *Reformat HDRs and confidence intervals objects*

Description

This function reformats HDRs and confidence intervals back and forth between set format and data frame format. If the object is a 'hdr' object (HDR presented as a set) it is reformatted into a 'hdr.df' object (HDR presented as a data frame) and *vice versa*. If the object is a 'ci' object (confidence interval as a set) it is reformatted into a 'ci.df' object (confidence interval presented as a data frame) and *vice versa*. All attributes and information is preserved when changing formats. If the object is not of a recognised kind (or is of multiple recognised kinds) then it is returned unchanged and the function gives a warning.

Usage

```
reformat(OBJ)

## S3 method for class 'hdr'
as.data.frame(x, ...)

## S3 method for class 'ci'
as.data.frame(x, ...)
```

Arguments

`OBJ`	An object to reformat (either a HDR or a confidence interval)
`x`	an R object
`...`	unused

Value

Returns the reformatted object

Index

as.data.frame.ci *(reformat)*, 15
as.data.frame.hdr *(reformat)*, 15

CONF, 2

HDR, 4
HDR.bimodal *(HDR.monotone)*, 13
HDR.discrete, 12
HDR.discrete.unimodal *(HDR.monotone)*, 13
HDR.monotone, 13
HDR.unimodal *(HDR.monotone)*, 13

reformat, 15