Package ‘gk’

February 4, 2018

Type Package
Title g-and-k and g-and-h Distribution Functions
Version 0.5.1
Date 2018-02-04
Description Functions for the g-and-k and generalised g-and-h distributions.
License GPL-2
Imports Ecdat, lubridate, progress, grDevices, graphics, stats
Suggests testthat
ByteCompile true
URL https://github.com/dennisprangle/gk

BugReports https://github.com/dennisprangle/gk/issues

RoxygenNote 6.0.1

NeedsCompilation no
Author Dennis Prangle [aut, cre, cph]
Maintainer Dennis Prangle <dennis.prangle@gmail.com>

Repository CRAN
Date/Publication 2018-02-04 09:24:39 UTC

R topics documented:

abc ................................................................. 2
abc_batch .......................................................... 3
dgh ............................................................... 3
dgk ............................................................... 5
fdsa ............................................................... 6
fx ................................................................. 8
improper_uniform_log_density .................................. 8
isValid .......................................................... 9
mcmc ............................................................ 10
momentEstimates ............................................... 11
orderstats ...................................................... 12
Approximate Bayesian computation inference

Description

Approximate Bayesian computation (ABC) inference for the g-and-k or g-and-h distribution.

Usage

```r
abc(x, N, model = c("gk", "gh"), logB = FALSE, rprior, M,
    sumstats = c("all order statistics", "octiles", "moment estimates"),
    silent = FALSE)
```

Arguments

- **x**: Vector of observations.
- **N**: Number of iterations to perform.
- **model**: Whether to fit g-and-k or g-and-h model.
- **logB**: When true, the second parameter is log(B) rather than B.
- **rprior**: A function with single argument, n, which returns a matrix with n rows consisting of samples from the prior distribution for 4 parameters e.g. (A,B,g,k).
- **M**: Number of simulations to accept.
- **sumstats**: Which summary statistics to use.
- **silent**: When FALSE (the default) a progress bar is shown.

Details

This function performs approximate Bayesian inference for iid data from a g-and-k or g-and-h distribution, avoiding expensive density calculations. The algorithm samples many parameter vectors from the prior and simulates corresponding data from the model. The parameters are accepted or rejected based on how similar the simulations are to the observed data. Similarity is measured using weighted Euclidean distance between summary vectors of the simulations and observations. Several summaries can be used, including the complete order statistics or summaries based on octiles. In the latter case only the corresponding order statistics are simulated, speeding up the method.

Value

Matrix whose rows are accepted parameter estimates plus a column giving the ABC distances.

References

Examples

```r
set.seed(1)
x = rgk(10, A=3, B=1, g=2, k=0.5) # An unusually small dataset for fast execution of this example
rprior = function(n) { matrix(runif(4*n, 0, 10), ncol=4) }
abc(x, N=1E4, rprior=rprior, M=100)
```

---

**abc_batch**  
*Single batch of ABC*

---

**Description**

Interal function carrying out part of ABC inference calculations

**Usage**

```r
abc_batch(sobs, priorSims, simStats, M, v = NULL)
```

**Arguments**

- `sobs`: Vector of observed summary statistics.
- `priorSims`: Matrix whose rows are vector of parameters drawn from the prior.
- `simStats`: Function mapping a vector of parameters to summary statistics.
- `M`: How many simulations to accept in this batch.
- `v`: Optional vector of estimated variances for each summary statistic

**Details**

Outputs a list containing: 1) matrix whose rows are accepted parameter vectors, and a column of resulting distances 2) value of v. If v is not supplied then variances are estimated here and returned. (The idea is that this is done for the first batch, and these values are reused from then on.)

---

**dgh**  
*g-and-h distribution functions*

---

**Description**

Density, distribution function, quantile function and random generation for the generalised g-and-h distribution
Usage

dgh(x, A, B, g, h, c = 0.8, log = FALSE)

pgh(q, A, B, g, h, c = 0.8, zscale = FALSE)

qgh(p, A, B, g, h, c = 0.8)

rgh(n, A, B, g, h, c = 0.8)

Arguments

x Vector of quantiles.
A Vector of A (location) parameters.
B Vector of B (scale) parameters. Must be positive.
g Vector of g parameters.
h Vector of h parameters. Must be non-negative.
c Vector of c parameters. Often fixed at 0.8 which is the default.
log If true the log density is returned.
q Vector of quantiles.
zscale If true the N(0,1) quantile of the cdf is returned.
p Vector of probabilities.
n Number of draws to make.

Details

The (generalised) g-and-h distribution is defined by its quantile function:

\[ x(p) = A + B[1 + c \tanh(gz/2)]z \exp(hz^2/2), \]

where z is the standard normal quantile of p. Parameter restrictions include \( B > 0 \) and \( h \geq 0 \). Typically c=0.8. For more background information see the references.

rgh and qgh use quick direct calculations. However dgh and pgh involve slower numerical inversion of the quantile function.

Especially extreme values of the inputs will produce pgh output rounded to 0 or 1 (-Inf or Inf for zscale=TRUE). The corresponding dgh output will be 0 or -Inf for log=TRUE.

Value

dgh gives the density, pgh gives the distribution, qgh gives the quantile function, and rgh generates random deviates

References

Examples

\[ p = 1:9/10 \]  ##Some probabilities
\[ x = qgh(seq(0.1,0.9,0.1), A=3, B=1, g=2, h=0.5) \]  ##g-and-h quantiles
\[ rgh(5, A=3, B=1, g=2, h=0.5) \]  ##g-and-h draws
\[ dgh(x, A=3, B=1, g=2, h=0.5) \]  ##Densities of x under g-and-h
\[ dgh(x, A=3, B=1, g=2, h=0.5, log=TRUE) \]  ##Log densities of x under g-and-h
\[ pgh(x, A=3, B=1, g=2, h=0.5) \]  ##Distribution function of x under g-and-h

---

dgk  

\textit{g-and-k distribution functions}

Description

Density, distribution function, quantile function and random generation for the g-and-k distribution.

Usage

\[ dgk(x, A, B, g, k, c = 0.8, \log = \text{FALSE}) \]
\[ pgk(q, A, B, g, k, c = 0.8, \text{zscale} = \text{FALSE}) \]
\[ qgk(p, A, B, g, k, c = 0.8) \]
\[ rgk(n, A, B, g, k, c = 0.8) \]

Arguments

- **x**: Vector of quantiles.
- **A**: Vector of A (location) parameters.
- **B**: Vector of B (scale) parameters. Must be positive.
- **g**: Vector of g parameters.
- **k**: Vector of k parameters. Must be at least -0.5.
- **c**: Vector of c parameters. Often fixed at 0.8 which is the default.
- **log**: If true the log density is returned.
- **q**: Vector of quantiles.
- **zscale**: If true the N(0,1) quantile of the cdf is returned.
- **p**: Vector of probabilities.
- **n**: Number of draws to make.
Details

The g-and-k distribution is defined by its quantile function:

\[ x(p) = A + B[1 + c \tanh(gz/2)]z(1 + z^2)^k, \]

where \( z \) is the standard normal quantile of \( p \). Parameter restrictions include \( B > 0 \) and \( k \geq -0.5 \). Typically \( c=0.8 \). For more background information see the references.

rgk and qgk use quick direct calculations. However dgk and pgk involve slower numerical inversion of the quantile function.

Especially extreme values of the inputs will produce pgk output rounded to 0 or 1 (-Inf or Inf for zscale=TRUE). The corresponding dgk output will be 0 or -Inf for log=TRUE.

Value

dgk gives the density, pgk gives the distribution, qgk gives the quantile function, and rgk generates random deviates

References


Examples

\[
\begin{align*}
p & = 1:9/10 \quad \#\text{Some probabilities} \\
x & = qgk(seq(0,1,0.9,0.1), A=3, B=1, g=2, k=0.5) \quad \#\text{g-and-k quantiles} \\
rk(5, A=3, B=1, g=2, k=0.5) \quad \#\text{g-and-k draws} \\
dgk(x, A=3, B=1, g=2, k=0.5) \quad \#\text{Densities of x under g-and-k} \\
dgk(x, A=3, B=1, g=2, k=0.5, log=TRUE) \quad \#\text{Log densities of x under g-and-k} \\
pgk(x, A=3, B=1, g=2, k=0.5) \quad \#\text{Distribution function of x under g-and-k}
\end{align*}
\]

fdsa

Finite difference stochastic approximation inference

Description

Finite difference stochastic approximation (FDSA) inference for the g-and-k or g-and-h distribution

Usage

\[
\text{fdsa}(x, N, \text{model} = \text{c("gk", "gh")}, \logB = \text{FALSE}, \text{theta0}, \text{batch}\_\text{size} = 100, \\
\text{alpha} = 1, \text{gamma} = 0.49, \text{a0} = 1, \text{c0} = \text{NULL}, \text{A} = 100, \\
\text{theta}\_\text{min} = \text{c(-Inf, ifelse(\logB, -Inf, 1e-05), -Inf, 0)}, \\
\text{theta}\_\text{max} = \text{c(Inf, Inf, Inf, Inf)}, \text{silent} = \text{FALSE}, \text{plot}\_\text{Every} = 100)
\]
Arguments

- **x**: Vector of observations.
- **N**: Number of iterations to perform.
- **model**: Whether to fit gk or gh model.
- **logB**: When true, the second parameter is log(B) rather than B.
- **theta0**: Vector of initial value for 4 parameters.
- **batch_size**: Mini-batch size.
- **alpha**: Gain decay for step size.
- **gamma**: Gain decay for finite difference.
- **a0**: Multiplicative step size tuning parameter (or vector of 4 values).
- **c0**: Multiplicative finite difference step tuning parameter (or vector of 4 values).
- **A**: Additive step size tuning parameter.
- **theta_min**: Vector of minimum values for each parameter.
- **theta_max**: Vector of maximum values for each parameter.
- **silent**: When FALSE (the default) a progress bar and intermediate results plots are shown.
- **plotEvery**: How often to plot the results if silent==FALSE.

Details

`fdsa` performs maximum likelihood inference for iid data from a g-and-k or g-and-h distribution, using simultaneous perturbation stochastic approximation. This should be faster than directly maximising the likelihood.

Value

Matrix whose rows are FDSA states: the initial state `theta0` and N subsequent states. The final row is the MLE estimate.

References


Examples

```r
set.seed(1)
x = rgk(10, A=3, B=1, g=2, k=0.5) ##An unusually small dataset for fast execution of this example
out = fdsa(x, N=100, theta0=c(mean(x),sd(x),0,0), theta_min=c(-5,1E-5,-5,0), theta_max=c(5,5,5,5))
```
**fx**

*Exchange rate example*

**Description**
Performs a demo analysis of exchange rate data

**Usage**

```r
fx(type = c("standard", "quick", "for paper"))
```

**Arguments**

- `type` What type of demo to perform. `standard` runs the full demo displaying plots. `quick` is a fast demo suitable for testing the package. `for paper` saves pdfs which can be used in the paper.

**Details**

`fx` performs the analysis of exchange rate data in the supporting reference (Prangle 2017).

**References**


**Examples**

```r
## Not run:
fx()
## End(Not run)
```

---

**improper_uniform_log_density**

*Improper uniform log density*

**Description**

Returns log density of an improper prior for the g-and-k or g-and-h distribution

**Usage**

```r
improper_uniform_log_density(theta)
```

**Arguments**

- `theta` A vector of 4 parameters representing (A,B,g,k) or (A,B,g,h)
isValid

Details

improper_uniform_log_density takes a 4 parameter vector as input and returns a log density value. The output corresponds to an improper uniform with constraints that the second and fourth parameters should be non-negative. These ensure that the resulting parameters are valid to use in the g-and-k or g-and-h distribution is valid. This function is supplied as a convenient default prior to use in the mcmc function.

Value

Value of an (unnormalised) log density

Examples

improper_uniform_log_density(c(0,1,0,0)) #Valid parameters - returns 0
improper_uniform_log_density(c(0,-1,0,0)) #Invalid parameters - returns -Inf

isInvalid

Check validity of g-and-k or g-and-h parameters

Description

Check whether parameter choices produce a valid g-and-k or g-and-h distribution.

Usage

isValid(g, k_or_h, c = 0.8, model = c("gk", "gh"), initial_z = seq(-1, 1, 0.2))

Arguments

g Vector of g parameters.
k_or_h Vector of k or h parameters.
c Vector of c parameters.
model Whether to check the g-and-k or g-and-h model.
initial_z Vector of initial z values to use in each optimisation.

Details

This function tests whether parameter choices provide a valid distribution. Only g k and c parameters need be supplied as A and B>0 have no effect. The function operates by numerically minimising the derivative of the quantile function, and returning TRUE if the minimum is positive. It is possible that a local minimum is found, so it is recommended to use multiple optimisation starting points, and to beware that false positive may still result!

Value

Logical vector denoting whether each parameter combination is valid.
mcmc

References


Examples

isValid(0:10, -0.5)
isValid(0:10, 0.5, c=0.9, model="gh")

Description

Markov chain Monte Carlo (MCMC) inference for the g-and-k or g-and-h distribution

Usage

mcmc(x, N, model = c("gk", "gh"), logB = FALSE,
get_log_prior = improper_uniform_log_density, theta0, Sigma0, t0 = 100,
epsilon = 1e-06, silent = FALSE, plotEvery = 100)

Arguments

x Vector of observations.
N Number of MCMC steps to perform.
model Whether to fit g-and-k or g-and-h model.
logB When true, the second parameter is log(B) rather than B.
get_log_prior A function with one argument (corresponding to a vector of 4 parameters e.g. A,B,g,k) returning the log prior density. This should ensure the parameters are valid - i.e. return -Inf for invalid parameters - as the mcmc code will not check this.
theta0 Vector of initial value for 4 parameters.
Sigma0 MCMC proposal covariance matrix
t0 Tuning parameter (number of initial iterations without adaptation).
epsilon Tuning parameter (weight given to identity matrix in covariance calculation).
silent When FALSE (the default) a progress bar and intermediate results plots are shown.
plotEvery How often to plot the results if silent==FALSE.

Details

mcmc performs Markov chain Monte Carlo inference for iid data from a g-and-k or g-and-h distribution, using the adaptive Metropolis algorithm of Haario et al (2001).
momentEstimates

Value

Matrix whose rows are MCMC states: the initial state theta0 and N subsequent states.

References


Examples

```r
set.seed(1)
x = rgk(10, A=3, B=1, g=2, k=0.5) ##An unusually small dataset for fast execution of this example
out = mcmc(x, N=1000, theta0=c(mean(x),sd(x),0,0), Sigma0=0.1*diag(4))
```

momentEstimates  Convert octiles to moment estimates

Description

Convert octiles to estimates of location, scale, skewness and kurtosis.

Usage

```r
momentEstimates(octiles)
```

Arguments

- `octiles`: Vector of octiles.

Details

Converts octiles to robust estimate of location, scale, skewness and kurtosis as used by Drovandi and Pettitt (2011).

Value

Vector of moment estimates.

References

orderstats

**Description**
Sample a subset of order statistics of the Uniform(0,1) distribution

**Usage**

\[
\text{orderstats}(n, \text{orderstats})
\]

**Arguments**

- \( n \) Total number of independent draws
- \( \text{orderstats} \) Which order statistics to generate, in increasing order

**Details**
Uniform order statistics are generated by the exponential spacings method (see Ripley for example).

**Value**
A vector of order statistics equal in length to \( \text{orderstats} \)

**References**
Brian Ripley ‘Stochastic Simulation’ Wiley (1987)

**Examples**

\[
\text{orderstats}(100, c(25, 50, 75))
\]
Index

abc, 2
abc_batch, 3
dgh, 3
dgk, 5
fdsa, 6
fx, 8
g-and-h (dgh), 3
g-and-k (dgk), 5
improper_uniform_log_density, 8
isValid, 9
mcmc, 10
momentEstimates, 11
orderstats, 12
pgh (dgh), 3
pgk (dgk), 5
qgh (dgh), 3
qgk (dgk), 5
rgh (dgh), 3
rgk (dgk), 5