

Package ‘GeneralizedHyperbolic’

May 15, 2018

Version 0.8-4

Date 2018-05-15

Title The Generalized Hyperbolic Distribution

Author David Scott <d.scott@auckland.ac.nz>

Maintainer David Scott <d.scott@auckland.ac.nz>

Depends R (>= 3.0.1)

Imports DistributionUtils, MASS

Suggests VarianceGamma, actuar, SkewHyperbolic, RUnit

Encoding latin1

Description Functions for the hyperbolic and related distributions.

Density, distribution and quantile functions and random number generation are provided for the hyperbolic distribution, the generalized hyperbolic distribution, the generalized inverse Gaussian distribution and the skew-Laplace distribution. Additional functionality is provided for the hyperbolic distribution, normal inverse Gaussian distribution and generalized inverse Gaussian distribution, including fitting of these distributions to data. Linear models with hyperbolic errors may be fitted using hyperblmFit.

License GPL (>= 2)

URL <https://r-forge.r-project.org/projects/rmetrics/>

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-15 14:38:15 UTC

R topics documented:

ArkansasRiver	3
Functions for Moments	4
Generalized Inverse Gaussian	5
GeneralizedHyperbolic	8
GeneralizedHyperbolicDistribution	10

GeneralizedHyperbolicPlots	13
ghypCalcRange	15
ghypChangePars	16
ghypCheckPars	18
ghypMom	19
ghypParam	21
ghypScale	22
gigCalcRange	23
gigChangePars	25
gigCheckPars	26
gigFit	27
gigFitStart	30
gigHessian	32
gigMom	33
gigParam	36
GIGPlots	37
hyperbCalcRange	38
hyperbChangePars	40
hyperbCvMTest	41
hyperbFit	43
hyperbFitStart	46
hyperbHessian	48
hyperblm	49
Hyperbolic	54
hyperbParam	57
HyperbPlots	58
hyperbWSqTable	59
mamquam	60
momRecursion	61
nervePulse	62
NIG	63
nigCalcRange	66
nigFit	68
nigFitStart	71
nigHessian	72
nigParam	74
nigPlots	75
plotShapeTriangle	76
resistors	77
SandP500	78
SkewLaplace	79
SkewLaplacePlots	80
Specific Generalized Hyperbolic Moments and Mode	82
Specific Generalized Inverse Gaussian Moments and Mode	83
Specific Hyperbolic Distribution Moments and Mode	85
Specific Normal Inverse Gaussian Distribution Moments and Mode	86
summary.gigFit	88
summary.hyperbFit	89

summary.hyperblm	91
summary.nigFit	93
traffic	95

Index	96
--------------	-----------

ArkansasRiver *Soil Electrical Conductivity*

Description

Electrical conductivity of soil paste extracts from the Lower Arkansas River Valley, at sites upstream and downstream of the John Martin Reservoir.

Usage

```
data(ArkansasRiver)
```

Format

The format is: List of 2 \$ upstream : num [1:823] 2.37 3.53 3.06 3.35 3.07 ... \$ downstream: num [1:435] 8.75 6.59 5.09 6.03 5.64 ...

Details

Electrical conductivity is a measure of soil water salinity.

Source

This data set was supplied by Eric Morway (<emorway@usgs.gov>).

References

Eric D. Morway and Timothy K. Gates (2011) Regional assessment of soil water salinity across an extensively irrigated river valley. *Journal of Irrigation and Drainage Engineering*, doi:10.1061/(ASCE)IR.1943-4774.0000411

Examples

```
data(ArkansasRiver)
lapply(ArkansasRiver, summary)
upstream <- ArkansasRiver[[1]]
downstream <- ArkansasRiver[[2]]
## Fit normal inverse Gaussian
## Hyperbolic can also be fitted but fit is not as good
fitUpstream <- nigFit(upstream)
summary(fitUpstream)
par(mfrow = c(2,2))
plot(fitUpstream)
fitDownstream <- nigFit(downstream)
```

```

summary(fitDownstream)
plot(fitDownstream)
par(mfrow = c(1,1))
## Combined plot to compare
## Reproduces Figure 3 from Morway and Gates (2011)
hist(upstream, col = "grey", xlab = "", ylab = "", cex.axis = 1.25,
     main = "", breaks = seq(0,20, by = 1), xlim = c(0,15), las = 1,
     ylim = c(0,0.5), freq = FALSE)
param <- coef(fitUpstream)
nigDens <- function(x) dnig(x, param = param)
curve(nigDens, 0, 15, n = 201, add = TRUE,
      ylab = NULL, col = "red", lty = 1, lwd = 1.7)

hist(downstream, add = TRUE, col = "black", angle = 45, density = 15,
     breaks = seq(0,20, by = 1), freq = FALSE)
param <- coef(fitDownstream)
nigDens <- function(x) dnig(x, param = param)
curve(nigDens, 0, 15, n = 201, add = TRUE,
      ylab = NULL, col = "red", lty = 1, lwd = 1.7)

mtext(expression(EC[e]), side = 1, line = 3, cex = 1.25)
mtext("Frequency", side = 2, line = 3, cex = 1.25)
legend(x = 7.5, y = 0.250, c("Upstream Region", "Downstream Region"),
      col = c("black", "black"), density = c(NA, 25),
      fill = c("grey", "black"), angle = c(NA, 45),
      cex = 1.25, bty = "n", xpd = TRUE)

```

Functions for Moments *Functions for Calculating Moments*

Description

Functions used to calculate the mean, variance, skewness and kurtosis of a hyperbolic distribution. Not expected to be called directly by users.

Usage

```

RLambda(zeta, lambda = 1)
SLambda(zeta, lambda = 1)
MLambda(zeta, lambda = 1)
WLambda1(zeta, lambda = 1)
WLambda2(zeta, lambda = 1)
WLambda3(zeta, lambda = 1)
WLambda4(zeta, lambda = 1)
gammaLambda1(hyperbPi, zeta, lambda = 1)
gammaLambda1(hyperbPi, zeta, lambda = 1)

```

Arguments

hyperbPi	Value of the parameter π of the hyperbolic distribution.
zeta	Value of the parameter ζ of the hyperbolic distribution.
lambda	Parameter related to order of Bessel functions.

Value

The functions `RLambda` and `SLambda` are used in the calculation of the mean and variance. They are functions of the Bessel functions of the third kind, implemented in R as `besselK`. The other functions are used in calculation of higher moments. See Barndorff-Nielsen, O. and Blæsild, P. (1981) for details of the calculations.

The parameterization of the hyperbolic distribution used for this and other components of the `HyperbolicDist` package is the (π, ζ) one. See `hyperbChangePars` to transfer between parameterizations.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Richard Trendall, Thomas Tran

References

Barndorff-Nielsen, O. and Blæsild, P (1981). Hyperbolic distributions and ramifications: contributions to theory and application. In *Statistical Distributions in Scientific Work*, eds., Taillie, C., Patil, G. P., and Baldessari, B. A., Vol. 4, pp. 19–44. Dordrecht: Reidel.

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

See Also

[dhyperb](#), [hyperbMean](#), [hyperbChangePars](#), [besselK](#)

Generalized Inverse Gaussian

Generalized Inverse Gaussian Distribution

Description

Density function, cumulative distribution function, quantile function and random number generation for the generalized inverse Gaussian distribution with parameter vector `param`. Utility routines are included for the derivative of the density function and to find suitable break points for use in determining the distribution function.

Usage

```

dgig(x, chi = 1, psi = 1, lambda = 1,
     param = c(chi, psi, lambda), KOmega = NULL)
pgig(q, chi = 1, psi = 1, lambda = 1,
     param = c(chi, psi, lambda), lower.tail = TRUE,
     ibfTol = .Machine$double.eps^(0.85), nmax = 200)
qgig(p, chi = 1, psi = 1, lambda = 1,
     param = c(chi, psi, lambda), lower.tail = TRUE,
     method = c("spline", "integrate"),
     nInterpol = 501, uniTol = 10^(-7),
     ibfTol = .Machine$double.eps^(0.85), nmax = 200, ...)
rgig(n, chi = 1, psi = 1, lambda = 1,
     param = c(chi, psi, lambda))
rgig1(n, chi = 1, psi = 1, param = c(chi, psi))
ddgig(x, chi = 1, psi = 1, lambda = 1,
      param = c(chi, psi, lambda), KOmega = NULL)

```

Arguments

<code>x, q</code>	Vector of quantiles.
<code>p</code>	Vector of probabilities.
<code>n</code>	Number of observations to be generated.
<code>chi</code>	A shape parameter that by default holds a value of 1.
<code>psi</code>	Another shape parameter that is set to 1 by default.
<code>lambda</code>	Shape parameter of the GIG distribution. Common to all forms of parameterization. By default this is set to 1.
<code>param</code>	Parameter vector taking the form <code>c(chi, psi, lambda)</code> for <code>rgig</code> , or <code>c(chi, psi)</code> for <code>rgig1</code> .
<code>method</code>	Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
<code>lower.tail</code>	Logical. If TRUE, probabilities are $P(X \leq x)$, otherwise as $P(X > x)$.
<code>KOmega</code>	Sets the value of the Bessel function in the density or derivative of the density. See Details .
<code>ibfTol</code>	Value of tolerance to be passed to <code>incompleteBesselK</code> by <code>pgig</code> .
<code>nmax</code>	Value of maximum order of the approximating series to be passed to <code>incompleteBesselK</code> by <code>pgig</code> .
<code>nInterpol</code>	The number of points used in <code>qgig</code> for cubic spline interpolation (see <code>splinefun</code>) of the distribution function.
<code>uniTol</code>	Value of <code>tol</code> in calls to <code>uniroot</code> . See <code>uniroot</code> .
<code>...</code>	Passes arguments to <code>uniroot</code> . See Details .

Details

The generalized inverse Gaussian distribution has density

$$f(x) = \frac{(\psi/\chi)^{\frac{\lambda}{2}}}{2K_{\lambda}(\sqrt{\psi\chi})} x^{\lambda-1} e^{-\frac{1}{2}(\chi x^{-1} + \psi x)}$$

for $x > 0$, where $K_{\lambda}()$ is the modified Bessel function of the third kind with order λ .

The generalized inverse Gaussian distribution is investigated in detail in Jörgensen (1982).

Use `gigChangePars` to convert from the (δ, γ) , (α, β) , or (ω, η) parameterizations to the (χ, ψ) , parameterization used above.

`pgig` calls the function `incompleteBesselK` from the package **DistributionUtils** to integrate the density function `dgig`. This can be expected to be accurate to about 13 decimal places on a 32-bit computer, often more accurate. The algorithm used is due to Slavinsky and Safouhi (2010).

Calculation of quantiles using `qgig` permits the use of two different methods. Both methods use `uniroot` to find the value of x for which a given q is equal $F(x)$ where F denotes the cumulative distribution function. The difference is in how the numerical approximation to F is obtained. The obvious and more accurate method is to calculate the value of $F(x)$ whenever it is required using a call to `pghyp`. This is what is done if the method is specified as "integrate". It is clear that the time required for this approach is roughly linear in the number of quantiles being calculated. A Q-Q plot of a large data set will clearly take some time. The alternative (and default) method is that for the major part of the distribution a spline approximation to $F(x)$ is calculated and quantiles found using `uniroot` with this approximation. For extreme values (for which the tail probability is less than 10^{-7}), the integration method is still used even when the method specified is "spline".

If accurate probabilities or quantiles are required, tolerances (`intTol` and `uniTol`) should be set to small values, say 10^{-10} or 10^{-12} with method = "integrate". Generally then accuracy might be expected to be at least 10^{-9} . If the default values of the functions are used, accuracy can only be expected to be around 10^{-4} . Note that on 32-bit systems `.Machine$double.eps^0.25 = 0.0001220703` is a typical value.

Generalized inverse Gaussian observations are obtained via the algorithm of Dagpunar (1989).

Value

`dgig` gives the density, `pgig` gives the distribution function, `qgig` gives the quantile function, and `rgig` generates random variates. `rgig1` generates random variates in the special case where $\lambda = 1$. `ddgig` gives the derivative of `dgig`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Richard Trendall, and Melanie Luen.

References

- Dagpunar, J.S. (1989). An easily implemented generalised inverse Gaussian generator. *Commun. Statist. -Simula.*, **18**, 703–710.
- Jörgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

Slevinsky, Richard M., and Safouhi, Hassan (2010) A recursive algorithm for the G transformation and accurate computation of incomplete Bessel functions. *Appl. Numer. Math.*, In press.

See Also

[safeIntegrate](#), [integrate](#) for its shortfalls, [splinefun](#), [uniroot](#) and [gigChangePars](#) for changing parameters to the (χ, ψ) parameterization, [dghyp](#) for the generalized hyperbolic distribution.

Examples

```
param <- c(2, 3, 1)
gigRange <- gigCalcRange(param = param, tol = 10^(-3))
par(mfrow = c(1, 2))
curve(dgig(x, param = param), from = gigRange[1], to = gigRange[2],
      n = 1000)
title("Density of the\n Generalized Inverse Gaussian")
curve(pgig(x, param = param), from = gigRange[1], to = gigRange[2],
      n = 1000)
title("Distribution Function of the\n Generalized Inverse Gaussian")
dataVector <- rgig(500, param = param)
curve(dgig(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram\n of the Generalized Inverse Gaussian")
DistributionUtils::logHist(dataVector, main =
  "Log-Density and Log-Histogram\n of the Generalized Inverse Gaussian")
curve(log(dgig(x, param = param)), add = TRUE,
      range(dataVector)[1], range(dataVector)[2], n = 500)
par(mfrow = c(2, 1))
curve(dgig(x, param = param), from = gigRange[1], to = gigRange[2],
      n = 1000)
title("Density of the\n Generalized Inverse Gaussian")
curve(ddgig(x, param = param), from = gigRange[1], to = gigRange[2],
      n = 1000)
title("Derivative of the Density\n of the Generalized Inverse Gaussian")
```

GeneralizedHyperbolic *The Package ‘GeneralizedHyperbolic’: Summary Information*

Description

This package provides a collection of functions for working with the generalized hyperbolic and related distributions.

For the hyperbolic distribution functions are provided for the density function, distribution function, quantiles, random number generation and fitting the hyperbolic distribution to data (`hyperbFit`). The function `hyperbChangePars` will interchange parameter values between different parameterizations. The mean, variance, skewness, kurtosis and mode of a given hyperbolic distribution are given by `hyperbMean`, `hyperbVar`, `hyperbSkew`, `hyperbKurt`, and `hyperbMode` respectively. For assessing the fit of the hyperbolic distribution to a set of data, the log-histogram is useful. See

`logHist` from package **DistributionUtils**. Q-Q and P-P plots are also provided for assessing the fit of a hyperbolic distribution. A Cr amer-von-Mises test of the goodness of fit of data to a hyperbolic distribution is given by `hyperbCvMTest`. `S3 print`, `plot` and `summary` methods are provided for the output of `hyperbFit`.

For the generalized hyperbolic distribution functions are provided for the density function, distribution function, quantiles, and for random number generation. The function `ghypChangePars` will interchange parameter values between different parameterizations. The mean, variance, and mode of a given generalized hyperbolic distribution are given by `ghypMean`, `ghypVar`, `ghypSkew`, `ghypKurt`, and `ghypMode` respectively. Q-Q and P-P plots are also provided for assessing the fit of a generalized hyperbolic distribution.

For the generalized inverse Gaussian distribution functions are provided for the density function, distribution function, quantiles, and for random number generation. The function `gigChangePars` will interchange parameter values between different parameterizations. The mean, variance, skewness, kurtosis and mode of a given generalized inverse Gaussian distribution are given by `gigMean`, `gigVar`, `gigSkew`, `gigKurt`, and `gigMode` respectively. Q-Q and P-P plots are also provided for assessing the fit of a generalized inverse Gaussian distribution.

For the skew-Laplace distribution functions are provided for the density function, distribution function, quantiles, and for random number generation. Q-Q and P-P plots are also provided for assessing the fit of a skew-Laplace distribution.

Acknowledgements

A number of students have worked on the package: Ai-Wei Lee, Jennifer Tso, Richard Trendall, Thomas Tran, Simon Potter and David Cusack.

Thanks to Ross Ihaka and Paul Murrell for their willingness to answer my questions, and to all the core group for the development of R.

Special thanks also to Diethelm W urtz without whose advice, this package would be far inferior.

LICENCE

This package and its documentation are usable under the terms of the "GNU General Public License", a copy of which is distributed with the package.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.*, **A353**, 401–419.

Barndorff-Nielsen, O. and Bl esild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.*, **41**, 127–146.

Jørgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

GeneralizedHyperbolicDistribution

Generalized Hyperbolic Distribution

Description

Density function, distribution function, quantiles and random number generation for the generalized hyperbolic distribution, with parameters α (tail), β (skewness), δ (peakness), μ (location) and λ (shape).

Usage

```
dghyp(x, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda))
pghyp(q, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda),
      lower.tail = TRUE, subdivisions = 100,
      intTol = .Machine$double.eps^0.25, valueOnly = TRUE, ...)
qghyp(p, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda),
      lower.tail = TRUE, method = c("spline", "integrate"),
      nInterpol = 501, uniTol = .Machine$double.eps^0.25,
      subdivisions = 100, intTol = uniTol, ...)
rghyp(n, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda))
ddghyp(x, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
       param = c(mu, delta, alpha, beta, lambda))
```

Arguments

x, q	Vector of quantiles.
p	Vector of probabilities.
n	Number of random variates to be generated.
mu	Location parameter μ , default is 0.
delta	Scale parameter δ , default is 1.
alpha	Tail parameter α , default is 1.
beta	Skewness parameter β , default is 0.
lambda	Shape parameter λ , default is 1.
param	Specifying the parameters as a vector of the form <code>c(mu, delta, alpha, beta, lambda)</code> .

method	Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
lower.tail	Logical. If TRUE, probabilities are $P(X \leq x)$, otherwise they are $P(X > x)$.
subdivisions	The maximum number of subdivisions used to integrate the density and determine the accuracy of the distribution function calculation.
intTol	Value of rel.tol and hence abs.tol in calls to integrate. See integrate .
valueOnly	Logical. If valueOnly = TRUE calls to pghyp only return the value obtained for the integral. If valueOnly = FALSE an estimate of the accuracy of the numerical integration is also returned.
nInterpol	Number of points used in qghyp for cubic spline interpolation of the distribution function.
uniTol	Value of tol in calls to uniroot. See uniroot .
...	Passes additional arguments to integrate in pghyp and qghyp, and to uniroot in qghyp.

Details

Users may either specify the values of the parameters individually or as a vector. If both forms are specified, then the values specified by the vector param will overwrite the other ones. In addition the parameter values are examined by calling the function ghypCheckPars to see if they are valid.

The density function is

$$f(x) = c(\lambda, \alpha, \beta, \delta) \times \frac{K_{\lambda-1/2}(\alpha\sqrt{\delta^2 + (x-\mu)^2})}{\left(\frac{\sqrt{\delta^2 + (x-\mu)^2}}{\alpha}\right)^{1/2-\lambda}} e^{\beta(x-\mu)}$$

where $K_\nu(\cdot)$ is the modified Bessel function of the third kind with order ν , and

$$c(\lambda, \alpha, \beta, \delta) = \frac{\left(\frac{\sqrt{\alpha^2 - \beta^2}}{\delta}\right)^\lambda}{\sqrt{2\pi} K_\lambda(\delta\sqrt{\alpha^2 - \beta^2})}$$

Use ghypChangePars to convert from the (ρ, ζ) , (ξ, χ) , $(\bar{\alpha}, \bar{\beta})$, or (π, ζ) parameterizations to the (α, β) parameterization used above.

pghyp uses the function [integrate](#) to numerically integrate the density function. The integration is from $-\text{Inf}$ to x if x is to the left of the mode, and from x to Inf if x is to the right of the mode. The probability calculated this way is subtracted from 1 if required. Integration in this manner appears to make calculation of the quantile function more stable in extreme cases.

Calculation of quantiles using qghyp permits the use of two different methods. Both methods use uniroot to find the value of x for which a given q is equal $F(x)$ where F denotes the cumulative distribution function. The difference is in how the numerical approximation to F is obtained. The obvious and more accurate method is to calculate the value of $F(x)$ whenever it is required using a call to pghyp. This is what is done if the method is specified as "integrate". It is clear that the time required for this approach is roughly linear in the number of quantiles being calculated. A Q-Q plot of a large data set will clearly take some time. The alternative (and default) method is that for

the major part of the distribution a spline approximation to $F(x)$ is calculated and quantiles found using `uniroot` with this approximation. For extreme values (for which the tail probability is less than 10^{-7}), the integration method is still used even when the method specified is "spline".

If accurate probabilities or quantiles are required, tolerances (`intTol` and `uniTol`) should be set to small values, say 10^{-10} or 10^{-12} with `method = "integrate"`. Generally then accuracy might be expected to be at least 10^{-9} . If the default values of the functions are used, accuracy can only be expected to be around 10^{-4} . Note that on 32-bit systems `.Machine$double.eps^0.25 = 0.0001220703` is a typical value.

Value

`dghyp` gives the density function, `pghyp` gives the distribution function, `qghyp` gives the quantile function and `rghyp` generates random variates.

An estimate of the accuracy of the approximation to the distribution function can be found by setting `valueOnly = FALSE` in the call to `pghyp` which returns a list with components `value` and `error`.

`ddghyp` gives the derivative of `dghyp`.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Barndorff-Nielsen, O., and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S., and Read, C. B., Vol. 3, pp. 700–707.-New York: Wiley.

Bibby, B. M., and Sörensön,M. (2003). Hyperbolic processes in finance. In *Handbook of Heavy Tailed Distributions in Finance*,ed., Rachev, S. T. pp. 212–248. Elsevier Science B.~V.

Dagpunar, J.S. (1989). An easily implemented generalised inverse Gaussian generator *Commun. Statist.-Simula.*, **18**, 703–710.

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[dhyperb](#) for the hyperbolic distribution, [dgif](#) for the generalized inverse Gaussian distribution, [safeIntegrate](#), [integrate](#) for its shortfalls, also [splinefun](#), [uniroot](#) and [ghypChangePars](#) for changing parameters to the (α, β) parameterization.

Examples

```
param <- c(0, 1, 3, 1, 1/2)
ghypRange <- ghypCalcRange(param = param, tol = 10^(-3))
par(mfrow = c(1, 2))

### curves of density and distribution
curve(dghyp(x, param = param), ghypRange[1], ghypRange[2], n = 1000)
title("Density of the \n Generalized Hyperbolic Distribution")
```

```

curve(pghyp(x, param = param), ghypRange[1], ghypRange[2], n = 500)
title("Distribution Function of the \n Generalized Hyperbolic Distribution")

### curves of density and log density
par(mfrow = c(1, 2))
data <- rghyp(1000, param = param)
curve(dghyp(x, param = param), range(data)[1], range(data)[2],
      n = 1000, col = 2)
hist(data, freq = FALSE, add = TRUE)
title("Density and Histogram of the\n Generalized Hyperbolic Distribution")
DistributionUtils::logHist(data,
  main = "Log-Density and Log-Histogram of\n the Generalized Hyperbolic Distribution")
curve(log(dghyp(x, param = param)),
      range(data)[1], range(data)[2],
      n = 500, add = TRUE, col = 2)

### plots of density and derivative
par(mfrow = c(2, 1))
curve(dghyp(x, param = param), ghypRange[1], ghypRange[2], n = 1000)
title("Density of the\n Generalized Hyperbolic Distribution")
curve(ddghyp(x, param = param), ghypRange[1], ghypRange[2], n = 1000)
title("Derivative of the Density of the\n Generalized Hyperbolic Distribution")

```

GeneralizedHyperbolicPlots

Generalized Hyperbolic Quantile-Quantile and Percent-Percent Plots

Description

qqghyp produces a generalized hyperbolic Q-Q plot of the values in y .

ppghyp produces a generalized hyperbolic P-P (percent-percent) or probability plot of the values in y .

Graphical parameters may be given as arguments to qqghyp, and ppghyp.

Usage

```

qqghyp(y, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda),
      main = "Generalized Hyperbolic Q-Q Plot",
      xlab = "Theoretical Quantiles",
      ylab = "Sample Quantiles",
      plot.it = TRUE, line = TRUE, ...)

```

```

ppghyp(y, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
      param = c(mu, delta, alpha, beta, lambda),
      main = "Generalized Hyperbolic P-P Plot",
      xlab = "Uniform Quantiles",
      ylab = "Probability-integral-transformed Data",
      plot.it = TRUE, line = TRUE, ...)

```

Arguments

<code>y</code>	The data sample.
<code>mu</code>	μ is the location parameter. By default this is set to 0.
<code>delta</code>	δ is the scale parameter of the distribution. A default value of 1 has been set.
<code>alpha</code>	α is the tail parameter, with a default value of 1.
<code>beta</code>	β is the skewness parameter, by default this is 0.
<code>lambda</code>	λ is the shape parameter and dictates the shape that the distribution shall take. Default value is 1.
<code>param</code>	Parameters of the generalized hyperbolic distribution.
<code>xlab, ylab, main</code>	Plot labels.
<code>plot.it</code>	Logical. Should the result be plotted?
<code>line</code>	Add line through origin with unit slope.
<code>...</code>	Further graphical parameters.

Value

For `qqghyp` and `ppghyp`, a list with components:

<code>x</code>	The x coordinates of the points that are to be plotted.
<code>y</code>	The y coordinates of the points that are to be plotted.

References

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

See Also

[ppoints](#), [dghyp](#).

Examples

```
par(mfrow = c(1, 2))
y <- rghyp(200, param = c(2, 2, 2, 1, 2))
qqghyp(y, param = c(2, 2, 2, 1, 2), line = FALSE)
abline(0, 1, col = 2)
ppghyp(y, param = c(2, 2, 2, 1, 2))
```

 ghypCalcRange

Range of a Generalized Hyperbolic Distribution

Description

Given the parameter vector Theta of a generalized hyperbolic distribution, this function determines the range outside of which the density function is negligible, to a specified tolerance. The parameterization used is the (α, β) one (see [dghyp](#)). To use another parameterization, use [ghypChangePars](#).

Usage

```
ghypCalcRange(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
              param = c(mu, delta, alpha, beta, lambda),
              tol = 10^(-5), density = TRUE, ...)
```

Arguments

mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
lambda	λ is the shape parameter and dictates the shape that the distribution shall take. Default value is 1.
param	Value of parameter vector specifying the generalized hyperbolic distribution. This takes the form <code>c(mu, delta, alpha, beta, lambda)</code> .
tol	Tolerance.
density	Logical. If TRUE, the bounds are for the density function. If FALSE, they should be for the probability distribution, but this has not yet been implemented.
...	Extra arguments for calls to uniroot .

Details

The particular generalized hyperbolic distribution being considered is specified by the value of the parameter value `param`.

If `density = TRUE`, the function gives a range, outside of which the density is less than the given tolerance. Useful for plotting the density. Also used in determining break points for the separate sections over which numerical integration is used to determine the distribution function. The points are found by using [uniroot](#) on the density function.

If `density = FALSE`, the function returns the message: "Distribution function bounds not yet implemented".

Value

A two-component vector giving the lower and upper ends of the range.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

See Also

[dghyp](#), [ghypChangePars](#)

Examples

```
param <- c(0, 1, 5, 3, 1)
maxDens <- dghyp(ghypMode(param = param), param = param)
ghypRange <- ghypCalcRange(param = param, tol = 10^(-3) * maxDens)
ghypRange
curve(dghyp(x, param = param), ghypRange[1], ghypRange[2])
## Not run: ghypCalcRange(param = param, tol = 10^(-3), density = FALSE)
```

ghypChangePars

Change Parameterizations of the Generalized Hyperbolic Distribution

Description

This function interchanges between the following 5 parameterizations of the generalized hyperbolic distribution:

1. $\mu, \delta, \alpha, \beta, \lambda$
2. $\mu, \delta, \rho, \zeta, \lambda$
3. $\mu, \delta, \xi, \chi, \lambda$
4. $\mu, \delta, \bar{\alpha}, \bar{\beta}, \lambda$
5. $\mu, \delta, \pi, \zeta, \lambda$

The first four are the parameterizations given in Prause (1999). The final parameterization has proven useful in fitting.

Usage

```
ghypChangePars(from, to, param, noNames = FALSE)
```

Arguments

from	The set of parameters to change from.
to	The set of parameters to change to.
param	"from" parameter vector consisting of 5 numerical elements.
noNames	Logical. When TRUE, suppresses the parameter names in the output.

Details

In the 5 parameterizations, the following must be positive:

1. α, δ
2. ζ, δ
3. ξ, δ
4. $\bar{\alpha}, \delta$
5. ζ, δ

Furthermore, note that in the first parameterization α must be greater than the absolute value of β ; in the third parameterization, ξ must be less than one, and the absolute value of χ must be less than ξ ; and in the fourth parameterization, $\bar{\alpha}$ must be greater than the absolute value of $\bar{\beta}$.

Value

A numerical vector of length 5 representing param in the to parameterization.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Jennifer Tso, Richard Trendall

References

Barndorff-Nielsen, O. and Blæsild, P. (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[dghyp](#)

Examples

```
param1 <- c(0, 3, 2, 1, 2)           # Parameterization 1
param2 <- ghypChangePars(1, 2, param1) # Convert to parameterization 2
param2                                     # Parameterization 2
ghypChangePars(2, 1, param2)         # Back to parameterization 1
```

`ghypCheckPars`*Check Parameters of the Generalized Hyperbolic Distribution*

Description

Given a putative set of parameters for the generalized hyperbolic distribution, the functions checks if they are in the correct range, and if they correspond to the boundary cases.

Usage

```
ghypCheckPars(param)
```

Arguments

`param` Numeric. Putative parameter values for a generalized hyperbolic distribution.

Details

The vector `param` takes the form `c(mu, delta, alpha, beta, lambda)`.

If `alpha` is negative, an error is returned.

If `lambda` is 0 then the absolute value of `beta` must be less than `alpha` and `delta` must be greater than zero. If either of these conditions are false, then an error is returned.

If `lambda` is greater than 0 the absolute value of `beta` must be less than `alpha`. `delta` must also be non-negative. When either one of these is not true, an error is returned.

If `lambda` is less than 0 then the absolute value of `beta` must be equal to `alpha`. `delta` must be greater than 0, if both conditions are not true, an error is returned.

Value

A list with components:

`case` Either "" or "error".

`errorMessage` An appropriate error message if an error was found, the empty string "" otherwise.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Paoletta, Marc S. (2007) Intermediate Probability: A Computational Approach, Chichester: Wiley

See Also

[dghyp](#)

Examples

```

ghypCheckPars(c(0, 2.5, -0.5, 1, 0))      # error
ghypCheckPars(c(0, 2.5, 0.5, 0, 0))      # normal
ghypCheckPars(c(0, 1, 1, -1, 0))         # error
ghypCheckPars(c(2, 0, 1, 0.5, 0))        # error
ghypCheckPars(c(0, 5, 2, 1.5, 0))        # normal
ghypCheckPars(c(0, -2.5, -0.5, 1, 1))    # error
ghypCheckPars(c(0, -1, 0.5, 1, 1))       # error
ghypCheckPars(c(0, 0, -0.5, -1, 1))      # error
ghypCheckPars(c(2, 0, 0.5, 0, -1))       # error
ghypCheckPars(c(2, 0, 1, 0.5, 1))        # skew laplace
ghypCheckPars(c(0, 1, 1, 1, -1))         # skew hyperbolic

```

ghypMom

Calculate Moments of the Generalized Hyperbolic Distribution

Description

Function to calculate raw moments, mu moments, central moments and moments about any other given location for the generalized hyperbolic distribution.

Usage

```

ghypMom(order, mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
         param = c(mu, delta, alpha, beta, lambda),
         momType = c("raw", "central", "mu"), about = 0)

```

Arguments

order	Numeric. The order of the moment to be calculated. Not permitted to be a vector. Must be a positive whole number except for moments about zero.
mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
lambda	λ is the shape parameter and dictates the shape that the distribution shall take. Default value is 1.
param	Numeric. The parameter vector specifying the generalized hyperbolic distribution. Of the form <code>c(mu, delta, alpha, beta, lambda)</code> (see dghyp).
momType	Common types of moments to be calculated, default is "raw". See Details .
about	Numeric. The point around which the moment is to be calculated.

Details

Checking whether order is a whole number is carried out using the function [is.wholenumber](#).

momType can be either "raw" (moments about zero), "mu" (moments about mu), or "central" (moments about mean). If one of these moment types is specified, then there is no need to specify the about value. For moments about any other location, the about value must be specified. In the case that both momType and about are specified and contradicting, the function will always calculate the moments based on about rather than momType.

To calculate moments of the generalized hyperbolic distribution, the function firstly calculates mu moments by formula defined below and then transforms mu moments to central moments or raw moments or moments about any other locations as required by calling momChangeAbout.

The mu moments are obtained from the recursion formula given in Scott, Würtz and Tran (2011).

Value

The moment specified.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Scott, D. J., Würtz, D., Dong, C. and Tran, T. T. (2011) Moments of the generalized hyperbolic distribution. *Comp. Statistics.*, **26**, 459–476.

See Also

[ghypChangePars](#) and from package **DistributionUtils**: [logHist](#), [is.wholenumber](#), [momChangeAbout](#), and [momIntegrated](#).

Further, [ghypMean](#), [ghypVar](#), [ghypSkew](#), [ghypKurt](#).

Examples

```
param <- c(1, 2, 2, 1, 2)
mu <- param[1]
### mu moments
m1 <- ghypMean(param = param)
m1 - mu
ghypMom(1, param = param, momType = "mu")

## Comparison, using momIntegrated from pkg 'DistributionUtils':
momIntegrated <- DistributionUtils :: momIntegrated

momIntegrated("ghyp", order = 1, param = param, about = mu)
ghypMom(2, param = param, momType = "mu")
momIntegrated("ghyp", order = 2, param = param, about = mu)
ghypMom(10, param = param, momType = "mu")
momIntegrated("ghyp", order = 10, param = param, about = mu)
```

```

### raw moments
ghypMean(param = param)
ghypMom(1, param = param, momType = "raw")
momIntegrated("ghyp", order = 1, param = param, about = 0)
ghypMom(2, param = param, momType = "raw")
momIntegrated("ghyp", order = 2, param = param, about = 0)
ghypMom(10, param = param, momType = "raw")
momIntegrated("ghyp", order = 10, param = param, about = 0)

### central moments
ghypMom(1, param = param, momType = "central")
momIntegrated("ghyp", order = 1, param = param, about = m1)
ghypVar(param = param)
ghypMom(2, param = param, momType = "central")
momIntegrated("ghyp", order = 2, param = param, about = m1)
ghypMom(10, param = param, momType = "central")
momIntegrated("ghyp", order = 10, param = param, about = m1)

```

ghypParam

Parameter Sets for the Generalized Hyperbolic Distribution

Description

These objects store different parameter sets of the generalized hyperbolic distribution as matrices for testing or demonstration purposes.

The parameter sets `ghypSmallShape` and `ghypLargeShape` have a constant location parameter of $\mu = 0$, and constant scale parameter $\delta = 1$. In `ghypSmallParam` and `ghypLargeParam` the values of the location and scale parameters vary. In these parameter sets the location parameter $\mu = 0$ takes values from $\{0, 1\}$ and $\{-1, 0, 1, 2\}$ respectively. For the scale parameter δ , values are drawn from $\{1, 5\}$ and $\{1, 2, 5, 10\}$ respectively.

For the shape parameters α and β the approach is more complex. The values for these shape parameters were chosen by choosing values of ξ and χ which range over the shape triangle, then the function `ghypChangePars` was applied to convert them to the α, β parameterization. The resulting α, β values were then rounded to three decimal places. See the examples for the values of ξ and χ for the large parameter sets.

The values of λ are drawn from $\{-0.5, 0, 1\}$ in `ghypSmallShape` and $\{-1, -0.5, 0, 0.5, 1, 2\}$ in `ghypLargeShape`.

Usage

```

ghypSmallShape
ghypLargeShape
ghypSmallParam
ghypLargeParam

```

Format

`ghypSmallShape`: a 22 by 5 matrix; `ghypLargeShape`: a 90 by 5 matrix; `ghypSmallParam`: a 84 by 5 matrix; `ghypLargeParam`: a 1440 by 5 matrix.

Author(s)

David Scott <d.scott@auckland.ac.nz>

Examples

```
data(ghypParam)
plotShapeTriangle()
xis <- rep(c(0.1,0.3,0.5,0.7,0.9), 1:5)
chis <- c(0,-0.25,0.25,-0.45,0,0.45,-0.65,-0.3,0.3,0.65,
         -0.85,-0.4,0,0.4,0.85)
points(chis, xis, pch = 20, col = "red")

## Testing the accuracy of ghypMean
for (i in 1:nrow(ghypSmallParam)) {
  param <- ghypSmallParam[i, ]
  x <- rghyp(1000, param = param)
  sampleMean <- mean(x)
  funMean <- ghypMean(param = param)
  difference <- abs(sampleMean - funMean)
  print(difference)
}
```

ghypScale

Rescale a generalized hyperbolic distribution

Description

Given a specific mean and standard deviation will rescale any given generalized hyperbolic distribution to have the same shape but the specified mean and standard deviation. Can be used to standardize a generalized hyperbolic distribution to have mean zero and standard deviation one.

Usage

```
ghypScale(newMean, newSD,
          mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
          param = c(mu, delta, alpha, beta, lambda))
```

Arguments

newMean	Numeric. The required mean of the rescaled distribution.
newSD	Numeric. The required standard deviation of the rescaled distribution.
mu	Numeric. Location parameter μ of the starting distribution, default is 0.
delta	Numeric. Scale parameter δ of the starting distribution, default is 1.
alpha	Numeric. Tail parameter α of the starting distribution, default is 1.
beta	Numeric. Skewness parameter β of the starting distribution, default is 0.

lambda	Numeric. Shape parameter λ of the starting distribution, default is 1.
param	Numeric. Specifying the parameters of the starting distribution as a vector of the form <code>c(mu,delta,alpha,beta,lambda)</code> .

Value

A numerical vector of length 5 giving the value of the parameters in the rescaled generalized hyperbolic distribution in the usual (α, β) parameterization.

Author(s)

David Scott <d.scott@auckland.ac.nz>

Examples

```
param <- c(2,10,0.1,0.07,-0.5) # a normal inverse Gaussian
ghypMean(param = param)
ghypVar(param = param)
## convert to standardized parameters
(newParam <- ghypScale(0, 1, param = param))
ghypMean(param = newParam)
ghypVar(param = newParam)

## try some other mean and sd
(newParam <- ghypScale(1, 1, param = param))
ghypMean(param = newParam)
sqrt(ghypVar(param = newParam))
(newParam <- ghypScale(10, 2, param = param))
ghypMean(param = newParam)
sqrt(ghypVar(param = newParam))
```

gigCalcRange

Range of a Generalized Inverse Gaussian Distribution

Description

Given the parameter vector `param` of a generalized inverse Gaussian distribution, this function determines the range outside of which the density function is negligible, to a specified tolerance. The parameterization used is the (χ, ψ) one (see [dgig](#)). To use another parameterization, use [gigChangePars](#).

Usage

```
gigCalcRange(chi = 1, psi = 1, lambda = 1,
             param = c(chi, psi, lambda),
             tol = 10^(-5), density = TRUE, ...)
```

Arguments

chi	A shape parameter that by default holds a value of 1.
psi	Another shape parameter that is set to 1 by default.
lambda	Shape parameter of the GIG distribution. Common to all forms of parameterization. By default this is set to 1.
param	Value of parameter vector specifying the generalized inverse Gaussian distribution.
tol	Tolerance.
density	Logical. If TRUE, the bounds are for the density function. If FALSE, they should be for the probability distribution, but this has not yet been implemented.
...	Extra arguments for calls to uniroot .

Details

The particular generalized inverse Gaussian distribution being considered is specified by the value of the parameter value `param`.

If `density = TRUE`, the function gives a range, outside of which the density is less than the given tolerance. Useful for plotting the density. Also used in determining break points for the separate sections over which numerical integration is used to determine the distribution function. The points are found by using [uniroot](#) on the density function.

If `density = FALSE`, the function returns the message: "Distribution function bounds not yet implemented".

Value

A two-component vector giving the lower and upper ends of the range.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Jørgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

See Also

[dgig](#), [gigChangePars](#)

Examples

```
param <- c(2.5, 0.5, 5)
maxDens <- dgig(gigMode(param = param), param = param)
gigRange <- gigCalcRange(param = param, tol = 10^(-3) * maxDens)
gigRange
curve(dgig(x, param = param), gigRange[1], gigRange[2])
## Not run: gigCalcRange(param = param, tol = 10^(-3), density = FALSE)
```

gigChangePars	<i>Change Parameterizations of the Generalized Inverse Gaussian Distribution</i>
---------------	--

Description

This function interchanges between the following 4 parameterizations of the generalized inverse Gaussian distribution:

1. (χ, ψ, λ)
2. $(\delta, \gamma, \lambda)$
3. (α, β, λ)
4. (ω, η, λ)

See Jörgensen (1982) and Dagpunar (1989)

Usage

```
gigChangePars(from, to, param, noNames = FALSE)
```

Arguments

from	The set of parameters to change from.
to	The set of parameters to change to.
param	“from” parameter vector consisting of 3 numerical elements.
noNames	Logical. When TRUE, suppresses the parameter names in the output.

Details

The range of λ is the whole real line. In each parameterization, the other two parameters must take positive values.

Value

A numerical vector of length 3 representing param in the “to” parameterization.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Jörgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

Dagpunar, J. S. (1989). An easily implemented generalised inverse Gaussian generator, *Commun. Statist.—Simula.*, **18**, 703–710.

See Also[dgig](#)**Examples**

```

param1 <- c(2.5, 0.5, 5)           # Parameterisation 1
param2 <- gigChangePars(1, 2, param1) # Convert to parameterization 2
param2           # Parameterization 2
gigChangePars(2, 1, as.numeric(param2)) # Convert back to parameterization 1

```

gigCheckPars*Check Parameters of the Generalized Inverse Gaussian Distribution*

Description

Given a putative set of parameters for the generalized inverse Gaussian distribution, the functions checks if they are in the correct range, and if they correspond to the boundary cases.

Usage

```
gigCheckPars(param, ...)
```

Arguments

<code>param</code>	Numeric. Putative parameter values for a generalized inverse Gaussian distribution.
<code>...</code>	Further arguments for calls to <code>all.equal</code> .

Details

The vector `param` takes the form `c(chi, psi, lambda)`.

If either `chi` or `psi` is negative, an error is returned.

If `chi` is 0 (to within tolerance allowed by `all.equal`) then `psi` and `lambda` must be positive or an error is returned. If these conditions are satisfied, the distribution is identified as a gamma distribution.

If `psi` is 0 (to within tolerance allowed by `all.equal`) then `chi` must be positive and `lambda` must be negative or an error is returned. If these conditions are satisfied, the distribution is identified as an inverse gamma distribution.

If both `chi` and `psi` are positive, then the distribution is identified as a normal generalized inverse Gaussian distribution.

Value

A list with components:

case	Whichever of "error", "gamma", invgamma, or "normal" is identified by the function.
errMessage	An appropriate error message if an error was found, the empty string "" otherwise.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Paolella, Marc S. (2007) Intermediate Probability: A Computational Approach, Chichester: Wiley

See Also

[dgig](#)

Examples

```
gigCheckPars(c(5, 2.5, -0.5)) # normal
gigCheckPars(c(-5, 2.5, 0.5)) # error
gigCheckPars(c(5, -2.5, 0.5)) # error
gigCheckPars(c(-5, -2.5, 0.5)) # error
gigCheckPars(c(0, 2.5, 0.5)) # gamma
gigCheckPars(c(0, 2.5, -0.5)) # error
gigCheckPars(c(0, 0, 0.5)) # error
gigCheckPars(c(0, 0, -0.5)) # error
gigCheckPars(c(5, 0, 0.5)) # error
gigCheckPars(c(5, 0, -0.5)) # invgamma
```

gigFit

Fit the Generalized Inverse Gaussain Distribution to Data

Description

Fits a generalized inverse Gaussian distribution to data. Displays the histogram, log-histogram (both with fitted densities), Q-Q plot and P-P plot for the fit which has the maximum likelihood.

Usage

```
gigFit(x, freq = NULL, paramStart = NULL,
       startMethod = c("Nelder-Mead", "BFGS"),
       startValues = c("LM", "GammaIG", "MoM", "Symb", "US"),
       method = c("Nelder-Mead", "BFGS", "nlm"),
       stand = TRUE, plots = FALSE, printOut = FALSE,
```

```

controlBFGS = list(maxit = 200),
controlNM = list(maxit = 1000),
maxitNLM = 1500, ...)

## S3 method for class 'gigFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'gigFit'
plot(x, which = 1:4,
     plotTitles = paste(c("Histogram of ", "Log-Histogram of ",
                          "Q-Q Plot of ", "P-P Plot of "),
                        x$obsName, sep = ""),
     ask = prod(par("mfcol")) < length(which) & dev.interactive(), ...)

## S3 method for class 'gigFit'
coef(object, ...)

## S3 method for class 'gigFit'
vcov(object, ...)

```

Arguments

<code>x</code>	Data vector for <code>gigFit</code> . Object of class "gigFit" for <code>print.gigFit</code> and <code>plot.gigFit</code> .
<code>freq</code>	A vector of weights with length equal to <code>length(x)</code> .
<code>paramStart</code>	A user specified starting parameter vector <code>param</code> taking the form <code>c(chi, psi, lambda)</code> .
<code>startMethod</code>	Method used by <code>gigFitStartMoM</code> in calls to <code>optim</code> .
<code>startValues</code>	Code giving the method of determining starting values for finding the maximum likelihood estimate of <code>param</code> .
<code>method</code>	Different optimisation methods to consider. See Details .
<code>stand</code>	Logical. If TRUE, the data is first standardized by dividing by the sample standard deviation.
<code>plots</code>	Logical. If FALSE suppresses printing of the histogram, log-histogram, Q-Q plot and P-P plot.
<code>printOut</code>	Logical. If FALSE suppresses printing of results of fitting.
<code>controlBFGS</code>	A list of control parameters for <code>optim</code> when using the "BFGS" optimisation.
<code>controlNM</code>	A list of control parameters for <code>optim</code> when using the "Nelder-Mead" optimisation.
<code>maxitNLM</code>	A positive integer specifying the maximum number of iterations when using the "nlm" optimisation.
<code>digits</code>	Desired number of digits when the object is printed.
<code>which</code>	If a subset of the plots is required, specify a subset of the numbers 1:4.
<code>plotTitles</code>	Titles to appear above the plots.

ask	Logical. If TRUE, the user is <i>asked</i> before each plot, see <code>par(ask = .)</code> .
...	Passes arguments to <code>optim</code> , <code>par</code> , <code>hist</code> , <code>logHist</code> , <code>qqgig</code> and <code>ppgig</code> .
object	Object of class "gigFit" for <code>coef.gigFit</code> and for <code>vcov.gigFit</code> .

Details

Possible values of the argument `startValues` are the following:

- "LM" Based on fitting linear models to the upper tails of the data x and the inverse of the data $1/x$.
- "GammaIG" Based on fitting gamma and inverse gamma distributions.
- "MoM" Method of moments.
- "Symb" Not yet implemented.
- "US" User-supplied.

If `startValues = "US"` then a value must be supplied for `paramStart`.

For the details concerning the use of `paramStart`, `startMethod`, and `startValues`, see `gigFitStart`.

The three optimisation methods currently available are:

- "BFGS" Uses the quasi-Newton method "BFGS" as documented in `optim`.
- "Nelder-Mead" Uses an implementation of the Nelder and Mead method as documented in `optim`.
- "nlm" Uses the `nlm` function in R.

For details of how to pass control information for optimisation using `optim` and `nlm`, see `optim` and `nlm`.

When `method = "nlm"` is used, warnings may be produced. These do not appear to be a problem.

Value

`gigFit` returns a list with components:

<code>param</code>	A vector giving the maximum likelihood estimate of <code>param</code> , as <code>c(chi, psi, lambda)</code> .
<code>maxLik</code>	The value of the maximised log-likelihood.
<code>method</code>	Optimisation method used.
<code>conv</code>	Convergence code. See the relevant documentation (either <code>optim</code> or <code>nlm</code>) for details on convergence.
<code>iter</code>	Number of iterations of optimisation routine.
<code>obs</code>	The data used to fit the generalized inverse Gaussian distribution.
<code>obsName</code>	A character string with the actual x argument name.
<code>paramStart</code>	Starting value of <code>param</code> returned by call to <code>gigFitStart</code> .
<code>svName</code>	Descriptive name for the method finding start values.
<code>startValues</code>	Acronym for the method of finding start values.
<code>breaks</code>	The cell boundaries found by a call to <code>hist</code> .
<code>midpoints</code>	The cell midpoints found by a call to <code>hist</code> .
<code>empDens</code>	The estimated density found by a call to <code>hist</code> .

Author(s)

David Scott <d.scott@auckland.ac.nz>, David Cusack

References

Jørgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

See Also

[optim](#), [par](#), [hist](#), [logHist](#) (pkg [DistributionUtils](#)), [qqgig](#), [ppgig](#), and [gigFitStart](#).

Examples

```
param <- c(1, 1, 1)
dataVector <- rgig(500, param = param)
## See how well gigFit works
gigFit(dataVector)
##gigFit(dataVector, plots = TRUE)

## See how well gigFit works in the limiting cases
## Gamma case
dataVector2 <- rgamma(500, shape = 1, rate = 1)
gigFit(dataVector2)

## Inverse gamma
require(actuar)
dataVector3 <- rinvgamma(500, shape = 1, rate = 1)
gigFit(dataVector3)

## Use nlm instead of default
gigFit(dataVector, method = "nlm")
```

`gigFitStart`

Find Starting Values for Fitting a Generalized Inverse Gaussian Distribution

Description

Finds starting values for input to a maximum likelihood routine for fitting the generalized inverse Gaussian distribution to data.

Usage

```
gigFitStart(x, startValues = c("LM", "GammaIG", "MoM", "Symb", "US"),
            paramStart = NULL,
            startMethodMoM = c("Nelder-Mead", "BFGS"), ...)
gigFitStartMoM(x, paramStart = NULL,
```

```

        startMethodMoM = "Nelder-Mead", ...)
gigFitStartLM(x, ...)

```

Arguments

x	Data vector.
startValues	Acronym indicating the method to use for obtaining starting values to be used as input to <code>gigFit</code> . See Details .
paramStart	Starting values for param if startValues = "US".
startMethodMoM	Method used by call to <code>optim</code> in finding method of moments estimates.
...	Passes arguments to <code>optim</code> and calls to <code>hist</code> .

Details

Possible values of the argument `startValues` are the following:

- "LM" Based on fitting linear models to the upper tails of the data x and the inverse of the data $1/x$.
- "GammaIG" Based on fitting gamma and inverse gamma distributions.
- "MoM" Method of moments.
- "Symb" Not yet implemented.
- "US" User-supplied.

If `startValues = "US"` then a value must be supplied for `paramStart`.

When `startValues = "MoM"` an initial optimisation is needed to find the starting values. This optimisations starts from arbitrary values, $c(1, 1, 1)$ for the parameters (χ, ψ, λ) and calls `optim` with the method given by `startMethodMoM`. Other starting values for the method of moments can be used by supplying a value for `paramStart`.

The default method of finding starting values is "LM". Testing indicates this is quite fast and finds good starting values. In addition, it does not require any starting values itself.

`gigFitStartMoM` is called by `gigFitStart` and implements the method of moments approach.

`gigFitStartLM` is called by `gigFitStart` and implements the linear models approach.

Value

`gigFitStart` returns a list with components:

paramStart	A vector with elements χ , ψ , and λ giving the starting value of param.
breaks	The cell boundaries found by a call to <code>hist</code> .
midpoints	The cell midpoints found by a call to <code>hist</code> .
empDens	The estimated density found by a call to <code>hist</code> .

`gigFitStartMoM` and `gigFitStartLM` each return `paramStart`, the starting value of param, to the calling function `gigFitStart`

Author(s)

David Scott <d.scott@auckland.ac.nz>, David Cusack

See Also

[dgig](#), [gigFit](#).

Examples

```
param <- c(1, 1, 1)
dataVector <- rgig(500, param = param)
gigFitStart(dataVector)
```

`gigHessian`

Calculate Two-Sided Hessian for the Generalized Inverse Gaussian Distribution

Description

Calculates the Hessian of a function, either exactly or approximately. Used to obtaining the information matrix for maximum likelihood estimation.

Usage

```
gigHessian(x, param, hessianMethod = "tsHessian",
           whichParam = 1)
```

Arguments

<code>x</code>	Data vector.
<code>param</code>	The maximum likelihood estimates parameter vector of the generalized inverse Gaussian distribution. There are five different sets of parameterizations can be used in this function, the first four sets are listed in <code>gigChangePars</code> and the last set is the log scale of the first set of the parameterization, i.e., $\mu, \log(\delta), \Pi, \log(\zeta)$.
<code>hessianMethod</code>	Only the approximate method (" <code>tsHessian</code> ") has actually been implemented so far.
<code>whichParam</code>	Numeric. A number between indicating which parameterization the argument <code>param</code> relates to. Only parameterization 1 is available so far.

Details

The approximate Hessian is obtained via a call to `tsHessian` from the package `DistributionUtils`. `summary.gigFit` calls the function `gigHessian` to calculate the Hessian matrix when the argument `hessian = TRUE`.

Value

`gigHessian` gives the approximate Hessian matrix for the data vector `x` and the estimated parameter vector `param`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, David Cusack

Examples

```
### Calculate the approximate Hessian using gigHessian:
param <- c(1,1,1)
dataVector <- rgig(500, param = param)
fit <- gigFit(dataVector)
coef <- coef(fit)
gigHessian(x = dataVector, param = coef, hessianMethod = "tsHessian",
           whichParam = 1)

### Or calculate the approximate Hessian using summary.gigFit method:
summary(fit, hessian = TRUE)
```

gigMom

Calculate Moments of the Generalized Inverse Gaussian Distribution

Description

Functions to calculate raw moments and moments about a given location for the generalized inverse Gaussian (GIG) distribution, including the gamma and inverse gamma distributions as special cases.

Usage

```
gigRawMom(order, chi = 1, psi = 1, lambda = 1,
           param = c(chi, psi, lambda))
gigMom(order, chi = 1, psi = 1, lambda = 1,
        param = c(chi, psi, lambda), about = 0)
gammaRawMom(order, shape = 1, rate = 1, scale = 1/rate)
```

Arguments

<code>order</code>	Numeric. The order of the moment to be calculated. Not permitted to be a vector. Must be a positive whole number except for moments about zero.
<code>chi</code>	A shape parameter that by default holds a value of 1.
<code>psi</code>	Another shape parameter that is set to 1 by default.
<code>lambda</code>	Shape parameter of the GIG distribution. Common to all forms of parameterization. By default this is set to 1.
<code>param</code>	Numeric. The parameter vector specifying the GIG distribution. Of the form <code>c(chi, psi, lambda)</code> (see dgig).

about	Numeric. The point around which the moment is to be calculated.
shape	Numeric. The shape parameter, must be non-negative, not permitted to be a vector.
scale	Numeric. The scale parameter, must be positive, not permitted to be a vector.
rate	Numeric. The rate parameter, an alternative way to specify the scale.

Details

The vector `param` of parameters is examined using `gigCheckPars` to see if the parameters are valid for the GIG distribution and if they correspond to the special cases which are the gamma and inverse gamma distributions. Checking of special cases and valid parameter vector values is carried out using the function `gigCheckPars`. Checking whether order is a whole number is carried out using the function `is.wholenumber`.

Raw moments (moments about zero) are calculated using the functions `gigRawMom` or `gammaRawMom`. For moments not about zero, the function `momChangeAbout` is used to derive moments about another point from raw moments. Note that raw moments of the inverse gamma distribution can be obtained from the raw moments of the gamma distribution because of the relationship between the two distributions. An alternative implementation of raw moments of the gamma and inverse gamma distributions may be found in the package **actuar** and these may be faster since they are written in C.

To calculate the raw moments of the GIG distribution it is convenient to use the alternative parameterization of the GIG in terms of ω and η , given as parameterization 3 in `gigChangePars`. Then the raw moment of the GIG distribution of order k is given by

$$\eta^k K_{\lambda+k}(\omega) / K_{\lambda}(\omega)$$

where $K_{\lambda}()$ is the modified Bessel function of the third kind of order λ .

The raw moment of the gamma distribution of order k with shape parameter α and rate parameter β is given by

$$\beta^{-k} \Gamma(\alpha + k) / \Gamma(\alpha)$$

The raw moment of order k of the inverse gamma distribution with shape parameter α and rate parameter β is the raw moment of order $-k$ of the gamma distribution with shape parameter α and rate parameter $1/\beta$.

Value

The moment specified. In the case of raw moments, `Inf` is returned if the moment is infinite.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley

See Also

[gigCheckPars](#), [gigChangePars](#) and from package **DistributionUtils**: [is.wholenumber](#), [momChangeAbout](#), [momIntegrated](#)

Further, [gigMean](#), [gigVar](#), [gigSkew](#), [gigKurt](#).

Examples

```
## Computations, using momIntegrated from pkg 'DistributionUtils':
momIntegrated <- DistributionUtils :: momIntegrated

### Raw moments of the generalized inverse Gaussian distribution
param <- c(5, 2.5, -0.5)
gigRawMom(1, param = param)
momIntegrated("gig", order = 1, param = param, about = 0)
gigRawMom(2, param = param)
momIntegrated("gig", order = 2, param = param, about = 0)
gigRawMom(10, param = param)
momIntegrated("gig", order = 10, param = param, about = 0)
gigRawMom(2.5, param = param)

### Moments of the generalized inverse Gaussian distribution
param <- c(5, 2.5, -0.5)
(m1 <- gigRawMom(1, param = param))
gigMom(1, param = param)
gigMom(2, param = param, about = m1)
(m2 <- momIntegrated("gig", order = 2, param = param, about = m1))
gigMom(1, param = param, about = m1)
gigMom(3, param = param, about = m1)
momIntegrated("gig", order = 3, param = param, about = m1)

### Raw moments of the gamma distribution
shape <- 2
rate <- 3
param <- c(shape, rate)
gammaRawMom(1, shape, rate)
momIntegrated("gamma", order = 1, shape = shape, rate = rate, about = 0)
gammaRawMom(2, shape, rate)
momIntegrated("gamma", order = 2, shape = shape, rate = rate, about = 0)
gammaRawMom(10, shape, rate)
momIntegrated("gamma", order = 10, shape = shape, rate = rate, about = 0)

### Moments of the inverse gamma distribution
param <- c(5, 0, -0.5)
gigRawMom(2, param = param)          # Inf
gigRawMom(-2, param = param)
momIntegrated("invgamma", order = -2, shape = -param[3],
             rate = param[1]/2, about = 0)

### An example where the moment is infinite: inverse gamma
param <- c(5, 0, -0.5)
gigMom(1, param = param)
gigMom(2, param = param)
```

gigParam

*Parameter Sets for the Generalized Inverse Gaussian Distribution***Description**

These objects store different parameter sets of the generalized inverse Gaussian distribution as matrices for testing or demonstration purposes.

The parameter sets `gigSmallParam` and `gigLargeParam` give combinations of values of the parameters χ , ψ and λ . For `gigSmallParam`, the values of χ and ψ are chosen from $\{0.1, 0.5, 2, 5, 20, 50\}$, and the values of λ from $\{-0.5, 0, 0.5, 1, 5\}$. For `gigLargeParam`, the values of χ and ψ are chosen from $\{0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}$, and the values of λ from $\{-2, -1, -0.5, 0, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$.

Usage

```
gigSmallParam
gigLargeParam
```

Format

`gigSmallParam`: a 125 by 3 matrix; `gigLargeParam`: a 1100 by 3 matrix.

Author(s)

David Scott <d.scott@auckland.ac.nz>

Examples

```
data(gigParam)
## Check values of chi and psi
plot(gigLargeParam[, 1], gigLargeParam[, 2])
### Check all three parameters
pairs(gigLargeParam,
      labels = c(expression(chi),expression(psi),expression(lambda)))

## Testing the accuracy of gigMean
for (i in 1:nrow(gigSmallParam)) {
  param <- gigSmallParam[i, ]
  x <- rgig(1000, param = param)
  sampleMean <- mean(x)
  funMean <- gigMean(param = param)
  difference <- abs(sampleMean - funMean)
  print(difference)
}
```

GIGPlots	<i>Generalized Inverse Gaussian Quantile-Quantile and Percent-Percent Plots</i>
----------	---

Description

qqgig produces a generalized inverse Gaussian QQ plot of the values in y .

ppgig produces a generalized inverse Gaussian PP (percent-percent) or probability plot of the values in y .

If `line = TRUE`, a line with zero intercept and unit slope is added to the plot.

Graphical parameters may be given as arguments to `qqgig`, and `ppgig`.

Usage

```
qqgig(y, chi = 1, psi = 1, lambda = 1,
      param = c(chi, psi, lambda),
      main = "GIG Q-Q Plot",
      xlab = "Theoretical Quantiles",
      ylab = "Sample Quantiles",
      plot.it = TRUE, line = TRUE, ...)
```

```
ppgig(y, chi = 1, psi = 1, lambda = 1,
      param = c(chi, psi, lambda),
      main = "GIG P-P Plot",
      xlab = "Uniform Quantiles",
      ylab = "Probability-integral-transformed Data",
      plot.it = TRUE, line = TRUE, ...)
```

Arguments

<code>y</code>	The data sample.
<code>chi</code>	A shape parameter that by default holds a value of 1.
<code>psi</code>	Another shape parameter that is set to 1 by default.
<code>lambda</code>	Shape parameter of the GIG distribution. Common to all forms of parameterization. By default this is set to 1.
<code>param</code>	Parameters of the generalized inverse Gaussian distribution.
<code>xlab, ylab, main</code>	Plot labels.
<code>plot.it</code>	Logical. TRUE denotes the results should be plotted.
<code>line</code>	Logical. If TRUE, a line with zero intercept and unit slope is added to the plot.
<code>...</code>	Further graphical parameters.

Value

For qqgig and ppgig, a list with components:

x The x coordinates of the points that are be plotted.
y The y coordinates of the points that are be plotted.

References

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

See Also

[ppoints](#), [dgig](#).

Examples

```
par(mfrow = c(1, 2))
y <- rgig(1000, param = c(2, 3, 1))
qqgig(y, param = c(2, 3, 1), line = FALSE)
abline(0, 1, col = 2)
ppgig(y, param = c(2, 3, 1))
```

hyperbCalcRange

Range of a Hyperbolic Distribution

Description

Given the parameter vector `param` of a hyperbolic distribution, this function calculates the range outside of which the distribution has negligible probability, or the density function is negligible, to a specified tolerance. The parameterization used is the (α, β) one (see [dhyperb](#)). To use another parameterization, use [hyperbChangePars](#).

Usage

```
hyperbCalcRange(mu = 0, delta = 1, alpha = 1, beta = 0,
                param = c(mu, delta, alpha, beta),
                tol = 10^(-5), density = TRUE, ...)
```

Arguments

mu μ is the location parameter. By default this is set to 0.
delta δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha α is the tail parameter, with a default value of 1.
beta β is the skewness parameter, by default this is 0.
param Value of parameter vector specifying the hyperbolic distribution. This takes the form `c(mu, delta, alpha, beta)`.

tol	Tolerance.
density	Logical. If FALSE, the bounds are for the probability distribution. If TRUE, they are for the density function.
...	Extra arguments for calls to uniroot .

Details

The particular hyperbolic distribution being considered is specified by the value of the parameter value `param`.

If `density = FALSE`, the function calculates the effective range of the distribution, which is used in calculating the distribution function and quantiles, and may be used in determining the range when plotting the distribution. By effective range is meant that the probability of an observation being greater than the upper end is less than the specified tolerance `tol`. Likewise for being smaller than the lower end of the range. Note that this has not been implemented yet.

If `density = TRUE`, the function gives a range, outside of which the density is less than the given tolerance. Useful for plotting the density.

Value

A two-component vector giving the lower and upper ends of the range.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Jennifer Tso, Richard Trendall

References

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

See Also

[dhyperb](#), [hyperbChangePars](#)

Examples

```
par(mfrow = c(1, 2))
param <- c(0, 1, 3, 1)
hyperbRange <- hyperbCalcRange(param = param, tol = 10^(-3))
hyperbRange
curve(phyperb(x, param = param), hyperbRange[1], hyperbRange[2])
maxDens <- dhyperb(hyperbMode(param = param), param = param)
hyperbRange <- hyperbCalcRange(param = param, tol = 10^(-3) * maxDens, density = TRUE)
hyperbRange
curve(dhyperb(x, param = param), hyperbRange[1], hyperbRange[2])
```

 hyperbChangePars

Change Parameterizations of the Hyperbolic Distribution

Description

This function interchanges between the following 4 parameterizations of the hyperbolic distribution:

1. μ, δ, π, ζ
2. $\mu, \delta, \alpha, \beta$
3. $\mu, \delta, \phi, \gamma$
4. μ, δ, ξ, χ

The first three are given in Barndorff-Nielsen and Blæsild (1983), and the fourth in Prause (1999)

Usage

```
hyperbChangePars(from, to, param, noNames = FALSE)
```

Arguments

from	The set of parameters to change from.
to	The set of parameters to change to.
param	"from" parameter vector consisting of 4 numerical elements.
noNames	Logical. When TRUE, suppresses the parameter names in the output.

Details

In the 4 parameterizations, the following must be positive:

1. ζ, δ
2. α, δ
3. ϕ, γ, δ
4. ξ, δ

Furthermore, note that in the second parameterization α must be greater than the absolute value of β , while in the fourth parameterization, ξ must be less than one, and the absolute value of χ must be less than ξ .

Value

A numerical vector of length 4 representing param in the to parameterization.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Jennifer Tso, Richard Trendall

References

Barndorff-Nielsen, O. and Blæsild, P. (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[dhyperb](#)

Examples

```
param1 <- c(2, 1, 3, 1)           # Parameterization 1
param2 <- hyperbChangePars(1, 2, param1) # Convert to parameterization 2
param2                               # Parameterization 2
hyperbChangePars(2, 1, param2)     # Back to parameterization 1
```

hyperbCvMTest

Cramer-von-Mises Test of a Hyperbolic Distribution

Description

Carry out a Crämer-von-Mises test of a hyperbolic distribution where the parameters of the distribution are estimated, or calculate the p-value for such a test.

Usage

```
hyperbCvMTest(x, mu = 0, delta = 1, alpha = 1, beta = 0,
              param = c(mu, delta, alpha, beta),
              conf.level = 0.95, ...)
hyperbCvMTestPValue(delta = 1, alpha = 1, beta = 0, Wsq, digits = 3)
## S3 method for class 'hyperbCvMTest'
print(x, prefix = "\t", ...)
```

Arguments

x	A numeric vector of data values for hyperbCvMTest, or object of class "hyperbCvMTest" for print.hyperbCvMTest.
mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
param	Parameters of the hyperbolic distribution taking the form c(mu, delta, alpha, beta).
conf.level	Confidence level of the the confidence interval.

...	Further arguments to be passed to or from methods.
Wsq	Value of the test statistic in the Cr�amer-von-Mises test of the hyperbolic distribution.
digits	Number of decimal places for p-value.
prefix	Character(s) to be printed before the description of the test.

Details

hyperbCvMTest carries out a Cr amer-von-Mises goodness-of-fit test of the hyperbolic distribution. The parameter `param` must be given in the (α, β) parameterization.

hyperbCvMTestPValue calculates the p-value of the test, and is not expected to be called by the user. The method used is interpolation in Table 5 given in Puig & Stephens (2001), which assumes all the parameters of the distribution are unknown. Since the table used is limited, large p-values are simply given as “>~0.25” and very small ones as “<~0.01”. The table is created as the matrix `wsqTable` when the package `GeneralizedHyperbolic` is invoked.

`print.hyperbCvMTest` prints the output from the Cr amer-von-Mises goodness-of-fit test for the hyperbolic distribution in very similar format to that provided by `print.htest`. The only reason for having a special print method is that p-values can be given as less than some value or greater than some value, such as “<\ ~0.01”, or “>\ ~0.25”.

Value

hyperbCvMTest returns a list with class `hyperbCvMTest` containing the following components:

<code>statistic</code>	The value of the test statistic.
<code>method</code>	A character string with the value “Cr�amer-von-Mises test of hyperbolic distribution”.
<code>data.name</code>	A character string giving the name(s) of the data.
<code>parameter</code>	The value of the parameter <code>param</code>
<code>p.value</code>	The p-value of the test.
<code>warn</code>	A warning if the parameter values are outside the limits of the table given in Puig & Stephens (2001).

hyperbCvMTestPValue returns a list with the elements `p.value` and `warn` only.

Author(s)

David Scott, Thomas Tran

References

Puig, Pedro and Stephens, Michael A. (2001), Goodness-of-fit tests for the hyperbolic distribution. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, **29**, 309–320.

Examples

```

param <- c(2, 2, 2, 1.5)
dataVector <- rhyperb(500, param = param)
fittedparam <- hyperbFit(dataVector)$param
hyperbCvMTest(dataVector, param = fittedparam)
dataVector <- rnorm(1000)
fittedparam <- hyperbFit(dataVector, startValues = "FN")$param
hyperbCvMTest(dataVector, param = fittedparam)

```

hyperbFit

Fit the Hyperbolic Distribution to Data

Description

Fits a hyperbolic distribution to data. Displays the histogram, log-histogram (both with fitted densities), Q-Q plot and P-P plot for the fit which has the maximum likelihood.

Usage

```

hyperbFit(x, freq = NULL, paramStart = NULL,
          startMethod = c("Nelder-Mead", "BFGS"),
          startValues = c("BN", "US", "FN", "SL", "MoM"),
          criterion = "MLE",
          method = c("Nelder-Mead", "BFGS", "nlm",
                    "L-BFGS-B", "nllminb", "constrOptim"),
          plots = FALSE, printOut = FALSE,
          controlBFGS = list(maxit = 200),
          controlNM = list(maxit = 1000), maxitNLM = 1500,
          controlLBFGSB = list(maxit = 200),
          controlNLMINB = list(),
          controlCO = list(), ...)

## S3 method for class 'hyperbFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'hyperbFit'
plot(x, which = 1:4,
     plotTitles = paste(c("Histogram of ", "Log-Histogram of ",
                          "Q-Q Plot of ", "P-P Plot of "), x$obsName,
                        sep = " "),
     ask = prod(par("mfcol")) < length(which) & dev.interactive(), ...)

## S3 method for class 'hyperbFit'
coef(object, ...)

## S3 method for class 'hyperbFit'
vcov(object, ...)

```

Arguments

<code>x</code>	Data vector for <code>hyperbFit</code> . Object of class "hyperbFit" for <code>print.hyperbFit</code> and <code>plot.hyperbFit</code> .
<code>freq</code>	A vector of weights with length equal to <code>length(x)</code> .
<code>paramStart</code>	A user specified starting parameter vector <code>param</code> taking the form <code>c(mu, delta, alpha, beta)</code> .
<code>startMethod</code>	Method used by <code>hyperbFitStart</code> in calls to <code>optim</code> .
<code>startValues</code>	Code giving the method of determining starting values for finding the maximum likelihood estimate of <code>param</code> .
<code>criterion</code>	Currently only "MLE" is implemented.
<code>method</code>	Different optimisation methods to consider. See Details .
<code>plots</code>	Logical. If FALSE suppresses printing of the histogram, log-histogram, Q-Q plot and P-P plot.
<code>printOut</code>	Logical. If FALSE suppresses printing of results of fitting.
<code>controlBFGS</code>	A list of control parameters for <code>optim</code> when using the "BFGS" optimisation.
<code>controlNM</code>	A list of control parameters for <code>optim</code> when using the "Nelder-Mead" optimisation.
<code>maxitNLM</code>	A positive integer specifying the maximum number of iterations when using the "nlm" optimisation.
<code>controlLBFGSB</code>	A list of control parameters for <code>optim</code> when using the "L-BFGS-B" optimisation.
<code>controlNLMINB</code>	A list of control parameters for <code>nlminb</code> when using the "nlminb" optimisation.
<code>controlCO</code>	A list of control parameters for <code>constrOptim</code> when using the "constrOptim" optimisation.
<code>digits</code>	Desired number of digits when the object is printed.
<code>which</code>	If a subset of the plots is required, specify a subset of the numbers 1:4.
<code>plotTitles</code>	Titles to appear above the plots.
<code>ask</code>	Logical. If TRUE, the user is <i>asked</i> before each plot, see <code>par(ask = .)</code> .
<code>...</code>	Passes arguments to <code>par</code> , <code>hist</code> , <code>logHist</code> , <code>qqhyperb</code> and <code>pphyperb</code> .
<code>object</code>	Object of class "hyperbFit" for <code>coef.hyperbFit</code> and for <code>vcov.hyperbFit</code> .

Details

`startMethod` can be either "BFGS" or "Nelder-Mead".

`startValues` can be one of the following:

- "US" User-supplied.
- "BN" Based on Barndorff-Nielsen (1977).
- "FN" A fitted normal distribution.
- "SL" Based on a fitted skew-Laplace distribution.
- "MoM" Method of moments.

For the details concerning the use of `paramStart`, `startMethod`, and `startValues`, see [hyperbFitStart](#).

The six optimisation methods currently available are:

- "BFGS" Uses the quasi-Newton method "BFGS" as documented in [optim](#).
- "Nelder-Mead" Uses an implementation of the Nelder and Mead method as documented in [optim](#).
- "nlm" Uses the [nlm](#) function in R.
- "L-BFGS-B" Uses the quasi-Newton method with box constraints "L-BFGS-B" as documented in [optim](#).
- "nlminb" Uses the [nlminb](#) function in R.
- "constrOptim" Uses the [constrOptim](#) function in R.

For details of how to pass control information for optimisation using `optim`, `nlm`, `nlminb` and `constrOptim`, see [optim](#), [nlm](#), [nlminb](#) and [constrOptim](#).

When `method = "nlm"` is used, warnings may be produced. These do not appear to be a problem.

Value

`hyperbFit` returns a list with components:

<code>param</code>	A vector giving the maximum likelihood estimate of <code>param</code> , as <code>c(mu, delta, alpha, beta)</code> .
<code>maxLik</code>	The value of the maximised log-likelihood.
<code>method</code>	Optimisation method used.
<code>conv</code>	Convergence code. See the relevant documentation (either optim or nlm) for details on convergence.
<code>iter</code>	Number of iterations of optimisation routine.
<code>obs</code>	The data used to fit the hyperbolic distribution.
<code>obsName</code>	A character string with the actual <code>x</code> argument name.
<code>paramStart</code>	Starting value of <code>param</code> returned by call to hyperbFitStart .
<code>svName</code>	Descriptive name for the method finding start values.
<code>startValues</code>	Acronym for the method of finding start values.
<code>breaks</code>	The cell boundaries found by a call to hist .
<code>midpoints</code>	The cell midpoints found by a call to hist .
<code>empDens</code>	The estimated density found by a call to hist .

Author(s)

David Scott <d.scott@auckland.ac.nz>, Ai-Wei Lee, Jennifer Tso, Richard Trendall, Thomas Tran, Christine Yang Dong

References

- Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.* **A353**, 401–419.
- Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.* **41**, 127–146.

See Also

[optim](#), [nlm](#), [nlminb](#), [constrOptim](#), [par](#), [hist](#), [logHist](#) (pkg **DistributionUtils**), [qqhyperb](#), [phyperb](#), [dskewlap](#) and [hyperbFitStart](#).

Examples

```
param <- c(2, 2, 2, 1)
dataVector <- rhyperb(500, param = param)
## See how well hyperbFit works
hyperbFit(dataVector)
hyperbFit(dataVector, plots = TRUE)
fit <- hyperbFit(dataVector)
par(mfrow = c(1, 2))
plot(fit, which = c(1, 3))

## Use nlm instead of default
hyperbFit(dataVector, method = "nlm")
```

hyperbFitStart

Find Starting Values for Fitting a Hyperbolic Distribution

Description

Finds starting values for input to a maximum likelihood routine for fitting hyperbolic distribution to data.

Usage

```
hyperbFitStart(x, startValues = c("BN", "US", "FN", "SL", "MoM"),
              paramStart = NULL,
              startMethodSL = c("Nelder-Mead", "BFGS"),
              startMethodMoM = c("Nelder-Mead", "BFGS"), ...)
hyperbFitStartMoM(x, startMethodMoM = "Nelder-Mead", ...)
```

Arguments

<code>x</code>	Data vector.
<code>startValues</code>	Vector of the different starting values to consider. See Details .
<code>paramStart</code>	Starting values for <code>param</code> if <code>startValues = "US"</code> .
<code>startMethodSL</code>	Method used by call to optim in finding skew Laplace estimates.
<code>startMethodMoM</code>	Method used by call to optim in finding method of moments estimates.
<code>...</code>	Passes arguments to hist and optim .

Details

Possible values of the argument `startValues` are the following:

- "US" User-supplied.
- "BN" Based on Barndorff-Nielsen (1977).
- "FN" A fitted normal distribution.
- "SL" Based on a fitted skew-Laplace distribution.
- "MoM" Method of moments.

If `startValues = "US"` then a value must be supplied for `paramStart`.

If `startValues = "MoM"`, `hyperbFitStartMoM` is called. These starting values are based on Barndorff-Nielsen *et al* (1985).

If `startValues = "SL"`, or `startValues = "MoM"` an initial optimisation is needed to find the starting values. These optimisations call `optim`.

Value

`hyperbFitStart` returns a list with components:

<code>paramStart</code>	A vector with elements <code>mu</code> , <code>delta</code> , <code>alpha</code> and <code>beta</code> giving the starting value of <code>param</code> .
<code>breaks</code>	The cell boundaries found by a call to <code>hist</code> .
<code>midpoints</code>	The cell midpoints found by a call to <code>hist</code> .
<code>empDens</code>	The estimated density found by a call to <code>hist</code> .

`hyperbFitStartMoM` returns only the method of moments estimates as a vector with elements `mu`, `delta`, `alpha` and `beta`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Ai-Wei Lee, Jennifer Tso, Richard Trendall, Thomas Tran

References

Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.*, **A353**, 401–419.

Barndorff-Nielsen, O., Blæsild, P., Jensen, J., and Sörenson, M. (1985). The fascination of sand. In *A celebration of statistics, The ISI Centenary Volume*, eds., Atkinson, A. C. and Fienberg, S. E., pp. 57–87. New York: Springer-Verlag.

Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.*, **41**, 127–146.

See Also

`dhyperb`, `dskewlap`, `hyperbFit`, `hist`, and `optim`.

Examples

```

param <- c(2, 2, 2, 1)
dataVector <- rhyperb(500, param = param)
hyperbFitStart(dataVector, startValues = "FN")
hyperbFitStartMoM(dataVector)
hyperbFitStart(dataVector, startValues = "MoM")

```

hyperbHessian

Calculate Two-Sided Hessian for the Hyperbolic Distribution

Description

Calculates the Hessian of a function, either exactly or approximately. Used to obtain the information matrix for maximum likelihood estimation.

Usage

```

hyperbHessian(x, param, hessianMethod = "exact",
              whichParam = 1:5)
sumX(x, mu, delta, r, k)

```

Arguments

x	Data vector.
param	The maximum likelihood estimates parameter vector of the hyperbolic distribution. There are five different sets of parameterizations can be used in this function, the first four sets are listed in hyperbChangePars and the last set is the log scale of the first set of the parameterization, i.e., $\mu, \log(\delta), \pi, \log(\zeta)$.
hessianMethod	Two methods are available to calculate the Hessian exactly ("exact") or approximately ("tsHessian").
whichParam	Numeric. A number between 1 to 5 indicating which set of the parameterization is the specified value in argument param belong to.
mu	Value of the parameter μ of the hyperbolic distribution.
delta	Value of the parameter δ of the hyperbolic distribution.
r	Parameter used in calculating a cumulative sum of the data vector x.
k	Parameter used in calculating a cumulative sum of the data vector x.

Details

The formulae for the exact Hessian are derived by Maple software with some simplifications. For now, the exact Hessian can only be obtained based on the first, second or the last parameterization sets. The approximate Hessian is obtained via a call to tsHessian from the package DistributionUtils. summary.hyperbFit calls the function hyperbHessian to calculate the Hessian matrix when the argument hessian = TRUE.

Value

hyperbHessian gives the approximate or exact Hessian matrix for the data vector x and the estimated parameter vector $param$. $sumX$ is a sum term used in calculating the exact Hessian. It is called by hyperbHessian when the argument `hessianMethod = "exact"`. It is not expected to be called directly by users.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

Examples

```
### Calculate the exact Hessian using hyperbHessian:
param <- c(2, 2, 2, 1)
dataVector <- rhyperb(500, param = param)
fit <- hyperbFit(dataVector, method = "BFGS")
coef <- coef(fit)
hyperbHessian(x = dataVector, param = coef, hessianMethod = "exact",
              whichParam = 2)

### Or calculate the exact Hessian using summary.hyperbFit method:
summary(fit, hessian = TRUE)

## Calculate the approximate Hessian:
summary(fit, hessian = TRUE, hessianMethod = "tsHessian")
```

 hyperblm

Fitting Linear Models with Hyperbolic Errors

Description

Fits linear models with hyperbolic errors. Can be used to carry out linear regression for data exhibiting heavy tails and skewness. Displays the histogram, log-histogram (both with fitted error distribution), Q-Q plot and residuals vs. fitted values plot for the fitted linear model.

Usage

```
hyperblm(formula, data, subset, weights, na.action,
          xx = FALSE, y = FALSE, contrasts = NULL,
          offset, method = "Nelder-Mead",
          startMethod = "Nelder-Mead", startStarts = "BN",
          paramStart = NULL,
          maxiter = 100, tolerance = 0.0001,
          controlBFGS = list(maxit = 1000),
          controlNM = list(maxit = 10000),
          maxitNLM = 10000,
          controlCO = list(), silent = TRUE, ...)
```

```
## S3 method for class 'hyperblm'
print(x, digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'hyperblm'
coef(object, ...)

## S3 method for class 'hyperblm'
plot(x, breaks = "FD",
      plotTitles = c("Residuals vs Fitted Values",
                     "Histogram of residuals",
                     "Log-Histogram of residuals",
                     "Q-Q Plot"),
      ...)
```

Arguments

formula	an object of class <code>"formula"</code> (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(w \cdot e^2)$); otherwise ordinary least squares is used. See also 'Details',
na.action	A function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The 'factory-fresh' default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
xx, y	Logicals. If <code>TRUE</code> , the corresponding components of the fit (the explanatory matrix and the response vector) are returned.
contrasts	An optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
offset	An optional vector. See Details .
method	Character. Possible values are <code>"BFGS"</code> , <code>"Nelder-Mead"</code> and <code>"nlm"</code> . See Details .
startMethod	Character. Possible values are <code>"BFGS"</code> and <code>"Nelder-Mead"</code> . See Details .
startStarts	Character. Possible values are <code>"BN"</code> , <code>"FN"</code> , <code>"SL"</code> , <code>"US"</code> and <code>"MoM"</code> . See Details .
paramStart	An optional vector. A vector of parameter start values for the optimization routine. See Details .
maxiter	Numeric. The maximum number of two-stage optimization alternating iterations. See Details .

tolerance	Numeric. The two-stage optimization convergence ratio. See Details .
controlBFGS, controlNLM	Lists. Lists of control parameters for <code>optim</code> when using corresponding (BFGS, Nelder-Mead) optimisation method in first stage. See <code>optim</code> .
maxitNLM	Numeric. The maximum number of iterations for the NLM optimizer.
controlCO	List. A list of control parameters for <code>constrOptim</code> in second stage.
silent	Logical. If TRUE, the error message of optimizer will not be displayed.
x	An object of class "hyperblm".
object	An object of class "hyperblm".
breaks	May be a vector, a single number or a character string. See <code>hist</code> .
plotTitles	Titles to appear above the plots.
digits	Numeric. Desired number of digits when the object is printed.
...	Passes additional arguments to function <code>hyperbFitStand</code> , <code>optim</code> and <code>constrOptim</code> .

Details

Models for `hyperblm` are specified symbolically. A typical model has the form $\text{response} \sim \text{terms}$ where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A `terms` specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first + second + first:second`.

If the formula includes an `offset`, this is evaluated and subtracted from the response.

If `response` is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See `model.matrix` for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on.

A formula has an implied intercept term. To remove this use either $y \sim x - 1$ or $y \sim 0 + x$. See `formula` for more details of allowed formulae.

Non-NULL weights can be used to indicate that different observations have different variances (with the values in `weights` being inversely proportional to the variances); or equivalently, when the elements of `weights` are positive integers w_i , that each response y_i is the mean of w_i unit-weight observations (including the case that there are w_i observations equal to y_i and the data have been summarized).

`hyperblm` calls the lower level function `hyperblmFit` for the actual numerical computations.

All of `weights`, `subset` and `offset` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

`hyperblmFit` uses a two-stage alternating optimization routine. The quality of parameter start values (especially the error distribution parameters) is crucial to the routine's convergence. The user can specify the start values via the `paramStart` argument, otherwise the function finds reliable start values by calling the `hyperbFitStand` function.

startMethod in the argument list is the optimization method for function `hyperbFitStandStart` which finds the start values for function `hyperbFitStand`. It is set to "Nelder-Mead" by default due to the robustness of this optimizer. The "BFGS" method is also implemented as it is relatively fast to converge. Since "BFGS" method is a quasi-Newton method it will not as robust and for some data will not achieve convergence.

startStarts is the method used to find the start values for function `hyperbFitStandStart` which includes:

- "BN" A method from Barndorff-Nielsen (1977) based on estimates of ψ and γ the absolute slopes of the left and right asymptotes to the log density function
- "FN" Based on a fitted normal distribution as it is a limit of the hyperbolic distribution
- "SL" Based on a fitted skew-Laplace distribution for which the log density has the form of two straight line with absolute slopes $1/\alpha$, $1/\beta$
- "MoM" A method of moment approach
- "US" User specified

method is the method used in stage one of the two-stage alternating optimization routine. As the startMethod, it is set to "Nelder-Mead" by default. Besides "BFGS", "nlm" is also implemented as an alternative. Since BFGS method is a quasi-Newton method it will not as robust and for some data will not achieve convergence.

If the maximum of the ratio the change of the individual coefficients is smaller than tolerance then the routine assumes convergence, otherwise if the alternating iteration number exceeds maxiter with the maximum of the ratio the change of the individual coefficients larger than tolerance, the routine is considered not to have converged.

Value

hyperblm returns an object of class "hyperblm" which is a list containing:

coefficients	A named vector of regression coefficients.
distributionParams	A named vector of fitted hyperbolic error distribution parameters.
fitted.values	The fitted values from the model.
residuals	The remainder after subtracting fitted values from response.
mle	The maximum likelihood value of the model.
method	The optimization method for stage one.
paramStart	The start values of parameters that the user specified (only where relevant).
residsParamStart	The start values of parameters obtained by hyperbFitStand (only where relevant).
call	The matched call.
terms	The terms object used.
contrasts	The contrasts used (only where relevant).
xlevels	The levels of the factors used in the fitting (only where relevant).

offset	The offset used (only where relevant)
xNames	The names of each explanatory variables. If explanatory variables don't have names then they will be named x.
yVec	The response vector.
xMatrix	The explanatory variables matrix.
iterations	Number of two-stage alternating iterations to convergence.
convergence	The convergence code for two stage optimization: 0 is the system converged, 1 is first stage does not converge, 2 is second stage does not converge, 3 is the both stages do not converge.
breaks	The cell boundaries found by a call the hist .

Author(s)

David Scott <d.scott@auckland.ac.nz>, Xinxing Li <xli053@aucklanduni.ac.nz>

References

- Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.*, **A353**, 401–419.
- Prause, K. (1999). *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.
- Trendall, Richard (2005). *hypReg: A Function for Fitting a Linear Regression Model in R with Hyperbolic Error*. Masters Thesis, Statistics Faculty, University of Auckland.
- Paoella, Marc S. (2007). *Intermediate Probability: A Computational Approach*. pp. 415 -Chichester: Wiley.
- Scott, David J. and Würtz, Diethelm and Chalabi, Yohan, (2011). *Fitting the Hyperbolic Distribution with R: A Case Study of Optimization Techniques*. In preparation.
- Stryhn, H. and Christensen, J. (2003). *Confidence intervals by the profile likelihood method, with applications in veterinary epidemiology*. ISVEE X.

See Also

[print.hyperblm](#) prints the regression result in a table. [coef.hyperblm](#) obtains the regression coefficients and error distribution parameters of the fitted model. [summary.hyperblm](#) obtains a summary output of class hyperblm object. [print.summary.hyperblm](#) prints the summary output in a table. [plot.hyperblm](#) obtains a residual vs fitted value plot, a histogram of residuals with error distribution density curve on top, a histogram of log residuals with error distribution error density curve on top and a QQ plot. [hyperblmFit](#), [optim](#), [nlm](#), [constrOptim](#), [hist](#), [hyperbFitStand](#), [hyperbFitStandStart](#).

Examples

```
### stackloss data example
## Not run:
airflow <- stackloss[, 1]
temperature <- stackloss[, 2]
```

```

acid <- stackloss[, 3]
stack <- stackloss[, 4]

hyperblm.fit <- hyperblm(stack ~ airflow + temperature + acid)

coef.hyperblm(hyperblm.fit)
plot.hyperblm(hyperblm.fit, breaks = 20)
summary.hyperblm(hyperblm.fit, hessian = FALSE)

## End(Not run)

```

Hyperbolic

Hyperbolic Distribution

Description

Density function, distribution function, quantiles and random number generation for the hyperbolic distribution with parameter vector `param`. Utility routines are included for the derivative of the density function and to find suitable break points for use in determining the distribution function.

Usage

```

dhyperb(x, mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))
phyperb(q, mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta),
        lower.tail = TRUE, subdivisions = 100,
        intTol = .Machine$double.eps^0.25,
        valueOnly = TRUE, ...)
qhyperb(p, mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta),
        lower.tail = TRUE, method = c("spline", "integrate"),
        nInterpol = 501, uniTol = .Machine$double.eps^0.25,
        subdivisions = 100, intTol = uniTol, ...)
rhyperb(n, mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))
ddhyperb(x, mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))

```

Arguments

<code>x, q</code>	Vector of quantiles.
<code>p</code>	Vector of probabilities.
<code>n</code>	Number of observations to be generated.
<code>mu</code>	μ is the location parameter. By default this is set to 0.
<code>delta</code>	δ is the scale parameter of the distribution. A default value of 1 has been set.

alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
param	Parameter vector taking the form <code>c(mu, delta, alpha, beta)</code> .
method	Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
lower.tail	Logical. If <code>lower.tail = TRUE</code> , the cumulative density is taken from the lower tail.
subdivisions	The maximum number of subdivisions used to integrate the density and determine the accuracy of the distribution function calculation.
intTol	Value of <code>rel.tol</code> and hence <code>abs.tol</code> in calls to <code>integrate</code> . See integrate .
valueOnly	Logical. If <code>valueOnly = TRUE</code> calls to <code>pghyp</code> only return the value obtained for the integral. If <code>valueOnly = FALSE</code> an estimate of the accuracy of the numerical integration is also returned.
nInterpol	Number of points used in <code>qghyp</code> for cubic spline interpolation of the distribution function.
uniTol	Value of <code>tol</code> in calls to <code>uniroot</code> . See uniroot .
...	Passes arguments to <code>uniroot</code> . See Details .

Details

The hyperbolic distribution has density

$$f(x) = \frac{1}{2\delta\sqrt{1+\pi^2}K_1(\zeta)} e^{-\zeta[\sqrt{1+\pi^2}\sqrt{1+(\frac{x-\mu}{\delta})^2} - \pi\frac{x-\mu}{\delta}]}$$

where $K_1()$ is the modified Bessel function of the third kind with order 1.

A succinct description of the hyperbolic distribution is given in Barndorff-Nielsen and Blæsild (1983). Three different possible parameterizations are described in that paper. A fourth parameterization is given in Prause (1999). All use location and scale parameters μ and δ . There are two other parameters in each case.

Use `hyperbChangePars` to convert from the (π, ζ) (ϕ, γ) or (ξ, χ) parameterizations to the (α, β) parameterization used above.

Each of the functions are wrapper functions for their equivalent generalized hyperbolic counterpart. For example, `dhyperb` calls `dghyp`. See [dghyp](#).

The hyperbolic distribution is a special case of the generalized hyperbolic distribution (Barndorff-Nielsen and Blæsild (1983)). The generalized hyperbolic distribution can be represented as a particular mixture of the normal distribution where the mixing distribution is the generalized inverse Gaussian. `rhyperb` uses this representation to generate observations from the hyperbolic distribution. Generalized inverse Gaussian observations are obtained via the algorithm of Dagpunar (1989).

Value

dhyperb gives the density, phyperb gives the distribution function, qhyperb gives the quantile function and rhyperb generates random variates. An estimate of the accuracy of the approximation to the distribution function may be found by setting `accuracy = TRUE` in the call to `phyperb` which then returns a list with components `value` and `error`.

ddhyperb gives the derivative of `dhyperb`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Ai-Wei Lee, Jennifer Tso, Richard Trendall

References

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

Dagpunar, J.S. (1989). An easily implemented generalized inverse Gaussian generator *Commun. Statist. -Simula.*, **18**, 703–710.

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[safeIntegrate](#), [integrate](#) for its shortfalls, [splinefun](#), [uniroot](#) and [hyperbChangePars](#) for changing parameters to the (α, β) parameterization, [dghyp](#) for the generalized hyperbolic distribution.

Examples

```
param <- c(0, 2, 1, 0)
hyperbRange <- hyperbCalcRange(param = param, tol = 10^(-3))
par(mfrow = c(1, 2))
curve(dhyperb(x, param = param), from = hyperbRange[1], to = hyperbRange[2],
      n = 1000)
title("Density of the\n Hyperbolic Distribution")
curve(phyperb(x, param = param), from = hyperbRange[1], to = hyperbRange[2],
      n = 1000)
title("Distribution Function of the\n Hyperbolic Distribution")
dataVector <- rhyperb(500, param = param)
curve(dhyperb(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram\n of the Hyperbolic Distribution")
DistributionUtils::logHist(dataVector, main =
  "Log-Density and Log-Histogram\n of the Hyperbolic Distribution")
curve(log(dhyperb(x, param = param)), add = TRUE,
      range(dataVector)[1], range(dataVector)[2], n = 500)
par(mfrow = c(2, 1))
curve(dhyperb(x, param = param), from = hyperbRange[1], to = hyperbRange[2],
      n = 1000)
```



```

title("Density of the\n Hyperbolic Distribution")
curve(ddhyperb(x, param = param), from = hyperbRange[1], to = hyperbRange[2],
      n = 1000)
title("Derivative of the Density\n of the Hyperbolic Distribution")

```

hyperbParam

Parameter Sets for the Hyperbolic Distribution

Description

These objects store different parameter sets of the hyperbolic distribution as matrices for testing or demonstration purposes.

The parameter sets `hyperbSmallShape` and `hyperbLargeShape` have a constant location parameter of $\mu = 0$, and constant scale parameter $\delta = 1$. In `hyperbSmallParam` and `hyperbLargeParam` the values of the location and scale parameters vary. In these parameter sets the location parameter $\mu = 0$ takes values from $\{0, 1\}$ and $\{-1, 0, 1, 2\}$ respectively. For the scale parameter δ , values are drawn from $\{1, 5\}$ and $\{1, 2, 5, 10\}$ respectively.

For the shape parameters α and β the approach is more complex. The values for these shape parameters were chosen by choosing values of ξ and χ which range over the shape triangle, then the function `hyperbChangePars` was applied to convert them to the α, β parameterization. The resulting α, β values were then rounded to three decimal places. See the examples for the values of ξ and χ for the large parameter sets.

Usage

```

hyperbSmallShape
hyperbLargeShape
hyperbSmallParam
hyperbLargeParam

```

Format

`hyperbSmallShape`: a 7 by 4 matrix; `hyperbLargeShape`: a 15 by 4 matrix; `hyperbSmallParam`: a 28 by 4 matrix; `hyperbLargeParam`: a 240 by 4 matrix.

Author(s)

David Scott <d.scott@auckland.ac.nz>

Examples

```

data(hyperbParam)
plotShapeTriangle()
xis <- rep(c(0.1,0.3,0.5,0.7,0.9), 1:5)
chis <- c(0,-0.25,0.25,-0.45,0,0.45,-0.65,-0.3,0.3,0.65,
        -0.85,-0.4,0,0.4,0.85)
points(chis, xis, pch = 20, col = "red")

```

```
## Testing the accuracy of hyperbMean
for (i in 1:nrow(hyperbSmallParam)) {
  param <- hyperbSmallParam[i, ]
  x <- rhyperb(1000, param = param)
  sampleMean <- mean(x)
  funMean <- hyperbMean(param = param)
  difference <- abs(sampleMean - funMean)
  print(difference)
}
```

 HyperbPlots

Hyperbolic Quantile-Quantile and Percent-Percent Plots

Description

qqhyperb produces a hyperbolic Q-Q plot of the values in y .

pphyperb produces a hyperbolic P-P (percent-percent) or probability plot of the values in y .

Graphical parameters may be given as arguments to qqhyperb, and pphyperb.

Usage

```
qqhyperb(y, mu = 0, delta = 1, alpha = 1, beta = 0,
  param = c(mu, delta, alpha, beta),
  main = "Hyperbolic Q-Q Plot",
  xlab = "Theoretical Quantiles",
  ylab = "Sample Quantiles",
  plot.it = TRUE, line = TRUE, ...)
```

```
pphyperb(y, mu = 0, delta = 1, alpha = 1, beta = 0,
  param = c(mu, delta, alpha, beta),
  main = "Hyperbolic P-P Plot",
  xlab = "Uniform Quantiles",
  ylab = "Probability-integral-transformed Data",
  plot.it = TRUE, line = TRUE, ...)
```

Arguments

y	The data sample.
μ	μ is the location parameter. By default this is set to 0.
δ	δ is the scale parameter of the distribution. A default value of 1 has been set.
α	α is the tail parameter, with a default value of 1.
β	β is the skewness parameter, by default this is 0.
param	Parameters of the hyperbolic distribution.

`xlab, ylab, main` Plot labels.
`plot.it` Logical. Should the result be plotted?
`line` Add line through origin with unit slope.
`...` Further graphical parameters.

Value

For `qqhyperb` and `phyperb`, a list with components:

`x` The x coordinates of the points that are to be plotted.
`y` The y coordinates of the points that are to be plotted.

References

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

See Also

[ppoints](#), [dhyperb](#), [hyperbFit](#)

Examples

```

par(mfrow = c(1, 2))
param <- c(2, 2, 2, 1.5)
y <- rhyperb(200, param = param)
qqhyperb(y, param = param, line = FALSE)
abline(0, 1, col = 2)
pphyperb(y, param = param)
  
```

hyperbWSqTable	<i>Percentage Points for the Cramer-von Mises Test of the Hyperbolic Distribution</i>
----------------	---

Description

This gives Table 5 of Puig & Stephens (2001) which is used for testing the goodness-of-fit of the hyperbolic distribution using the Cr amer-von-Mises test. It is for internal use by [hyperbCvMTest](#) and [hyperbCvMTestPValue](#) only and is not intended to be accessed by the user. It is loaded automatically when the package **HyperbolicDist** is invoked.

Usage

hyperbWSqTable

Format

The hyperbWsqTable matrix has 55 rows and 5 columns, giving percentage points of W^2 for different values of ξ and α (the rows), and of χ (the columns).

Source

Puig, Pedro and Stephens, Michael A. (2001), Goodness-of-fit tests for the hyperbolic distribution. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, **29**, 309–320.

 mamquam

Size of Gravels from Mamquam River

Description

Size of gravels collected from a sandbar in the Mamquam River, British Columbia, Canada. Summary data, giving the frequency of observations in 16 different size classes.

Usage

```
data(mamquam)
```

Format

The mamquam data frame has 16 rows and 2 columns.

[, 1]	midpoints	midpoints of intervals (psi units)
[, 2]	counts	number of observations in interval

Details

Gravel sizes are determined by passing clasts through templates of particular sizes. This gives a range in which the size of each clast lies. Sizes (in mm) are then converted into psi units by taking the base 2 logarithm of the size. The midpoints specified are the midpoints of the psi unit ranges, and counts gives the number of observations in each size range. The classes are of length 0.5 psi units. There are 3574 observations.

Source

Rice, Stephen and Church, Michael (1996) Sampling surficial gravels: the precision of size distribution percentile estimates. *J. of Sedimentary Research*, **66**, 654–665.

Examples

```
data(mamquam)
str(mamquam)
### Construct data from frequency summary, taking all observations
### at midpoints of intervals
```

```
psi <- rep(mamquam$midpoints, mamquam$counts)
barplot(table(psi))
### Fit the hyperbolic distribution
hyperbFit(psi)

### Actually hyperbFit can deal with frequency data
hyperbFit(mamquam$midpoints, freq = mamquam$counts)
```

momRecursion	<i>Computes the moment coefficients recursively for generalized hyperbolic and related distributions</i>
--------------	--

Description

This function computes all of the moments coefficients by recursion based on Scott, Würtz and Tran (2008). See **Details** for the formula.

Usage

```
momRecursion(order = 12, printMatrix = FALSE)
```

Arguments

order	Numeric. The order of the moment coefficients to be calculated. Not permitted to be a vector. Must be a positive whole number except for moments about zero.
printMatrix	Logical. Should the coefficients matrix be printed?

Details

The moment coefficients recursively as $a_{1,1} = 1$ and

$$a_{k,\ell} = a_{k-1,\ell-1} + (2\ell - k + 1)a_{k-1,\ell}$$

with $a_{k,\ell} = 0$ for $\ell < \lfloor (k+1)/2 \rfloor$ or $\ell > k$ where $k = \text{order}$, ℓ is equal to the integers from $(k+1)/2$ to k .

This formula is given in Scott, Würtz and Tran (2008, working paper).

The function also calculates M which is equal to $2\ell - k$. It is a common term which will appear in the formulae for calculating moments of generalized hyperbolic and related distributions.

Value

a	The non-zero moment coefficients for the specified order.
l	Integers from $(\text{order}+1)/2$ to order . It is used when computing the moment coefficients and the mu moments.
M	The common term used when computing mu moments for generalized hyperbolic and related distributions, $M = 2\ell - k$, $k=\text{order}$
lmin	The minimum of ℓ , which is equal to $(\text{order}+1)/2$.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

References

Scott, D. J., Würtz, D. and Tran, T. T. (2008) Moments of the Generalized Hyperbolic Distribution. Preprint.

Examples

```
momRecursion(order = 12)

#print out the matrix
momRecursion(order = 12, "true")
```

 nervePulse

Intervals Between Pulses Along a Nerve Fibre

Description

Times between successive electric pulses on the surface of isolated muscle fibres.

Usage

```
data(nervePulse)
```

Format

The nervePulse data is a vector with 799 observations.

Details

The end-plates of resting muscle fibres are the seat of spontaneous electric discharges. The occurrence of these spontaneous discharges at apparently normal synapses is studied in depth in Fatt and Katz (1951). The frequency and amplitude of these discharges was recorded. The times between each discharge were taken in milliseconds and this has been converted into the number of 1/50 sec intervals between successive pulses. There are 799 observations.

Source

Fatt, P., Katz, B. (1952) Spontaneous subthreshold activity at motor nerve endings. *J. of Physiology*, **117**, 109–128.

Jørgensen, B. (1982) Statistical Properties of the Generalized Inverse Gaussian Distribution. *Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York*

Examples

```

data(nervePulse)
str(nervePulse)

### Fit the generalized inverse Gaussian distribution
gigFit(nervePulse)

```

NIG

Normal Inverse Gaussian Distribution

Description

Density function, distribution function, quantiles and random number generation for the normal inverse Gaussian distribution with parameter vector `param`. Utility routines are included for the derivative of the density function and to find suitable break points for use in determining the distribution function.

Usage

```

dnig(x, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta))
pnig(q, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta),
      lower.tail = TRUE, subdivisions = 100,
      intTol = .Machine$double.eps^0.25, valueOnly = TRUE, ...)
qnig(p, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta),
      lower.tail = TRUE, method = c("spline", "integrate"),
      nInterpol = 501, uniTol = .Machine$double.eps^0.25,
      subdivisions = 100, intTol = uniTol, ...)
rnig(n, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta))
ddnig(x, mu = 0, delta = 1, alpha = 1, beta = 0,
       param = c(mu, delta, alpha, beta))

```

Arguments

<code>x, q</code>	Vector of quantiles.
<code>p</code>	Vector of probabilities.
<code>n</code>	Number of observations to be generated.
<code>mu</code>	μ is the location parameter. By default this is set to 0.
<code>delta</code>	δ is the scale parameter of the distribution. A default value of 1 has been set.
<code>alpha</code>	α is the tail parameter, with a default value of 1.
<code>beta</code>	β is the skewness parameter, by default this is 0.

param	Parameter vector taking the form <code>c(mu, delta, alpha, beta)</code> .
method	Character. If "spline" quantiles are found from a spline approximation to the distribution function. If "integrate", the distribution function used is always obtained by integration.
lower.tail	Logical. If <code>lower.tail = TRUE</code> , the cumulative density is taken from the lower tail.
subdivisions	The maximum number of subdivisions used to integrate the density and determine the accuracy of the distribution function calculation.
intTol	Value of <code>rel.tol</code> and hence <code>abs.tol</code> in calls to <code>integrate</code> . See integrate .
valueOnly	Logical. If <code>valueOnly = TRUE</code> calls to <code>pghyp</code> only return the value obtained for the integral. If <code>valueOnly = FALSE</code> an estimate of the accuracy of the numerical integration is also returned.
nInterpol	Number of points used in <code>qghyp</code> for cubic spline interpolation of the distribution function.
uniTol	Value of <code>tol</code> in calls to <code>uniroot</code> . See uniroot .
...	Passes arguments to <code>uniroot</code> . See Details .

Details

The normal inverse Gaussian distribution has density

$$e^{\delta\sqrt{\alpha^2-\beta^2}} \frac{\alpha\delta}{\pi\sqrt{\delta^2+(x-\mu)^2}} K_1(\alpha\sqrt{\delta^2+(x-\mu)^2}) e^{\beta(x-\mu)}$$

where $K_1()$ is the modified Bessel function of the third kind with order 1.

A succinct description of the normal inverse Gaussian distribution is given in Paoletta (2007). Because both of the normal inverse Gaussian distribution and the hyperbolic distribution are special cases of the generalized hyperbolic distribution (with different values of λ), the normal inverse Gaussian distribution has the same sets of parameterizations as the hyperbolic distribution. And therefore one can use `hyperbChangePars` to interchange between different parameterizations for the normal inverse Gaussian distribution as well (see `hyperbChangePars` for details).

Each of the functions are wrapper functions for their equivalent generalized hyperbolic distribution. For example, `dnig` calls `dghyp`.

`pnig` breaks the real line into eight regions in order to determine the integral of `dnig`. The break points determining the regions are found by `nigBreaks`, based on the values of `small`, `tiny`, and `deriv`. In the extreme tails of the distribution where the probability is `tiny` according to `nigCalcRange`, the probability is taken to be zero. In the range between where the probability is `tiny` and `small` according to `nigCalcRange`, an exponential approximation to the hyperbolic distribution is used. In the inner part of the distribution, the range is divided in 4 regions, 2 above the mode, and 2 below. On each side of the mode, the break point which forms the 2 regions is where the derivative of the density function is `deriv` times the maximum value of the derivative on that side of the mode. In each of the 4 inner regions the numerical integration routine [safeIntegrate](#) (which is a wrapper for [integrate](#)) is used to integrate the density `dnig`.

`qnig` uses the breakup of the real line into the same 8 regions as `pnig`. For quantiles which fall in the 2 extreme regions, the quantile is returned as `-Inf` or `Inf` as appropriate. In the range between

where the probability is tiny and small according to `nigCalcRange`, an exponential approximation to the hyperbolic distribution is used from which the quantile may be found in closed form. In the 4 inner regions `splinefun` is used to fit values of the distribution function generated by `pnig`. The quantiles are then found using the `uniroot` function.

`pnig` and `qnig` may generally be expected to be accurate to 5 decimal places.

Recall that the normal inverse Gaussian distribution is a special case of the generalized hyperbolic distribution and the generalized hyperbolic distribution can be represented as a particular mixture of the normal distribution where the mixing distribution is the generalized inverse Gaussian. `rnig` uses this representation to generate observations from the normal inverse Gaussian distribution. Generalized inverse Gaussian observations are obtained via the algorithm of Dagpunar (1989).

Value

`dnig` gives the density, `pnig` gives the distribution function, `qnig` gives the quantile function and `rnig` generates random variates. An estimate of the accuracy of the approximation to the distribution function may be found by setting `accuracy = TRUE` in the call to `pnig` which then returns a list with components `value` and `error`.

`ddnig` gives the derivative of `dnig`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong

References

- Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.
- Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley
- Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[safeIntegrate](#), [integrate](#) for its shortfalls, [splinefun](#), [uniroot](#) and [hyperbChangePars](#) for changing parameters to the (α, β) parameterization, [dghyp](#) for the generalized hyperbolic distribution.

Examples

```
param <- c(0, 2, 1, 0)
nigRange <- nigCalcRange(param = param, tol = 10^(-3))
par(mfrow = c(1, 2))
curve(dnig(x, param = param), from = nigRange[1], to = nigRange[2],
      n = 1000)
title("Density of the\n Normal Inverse Gaussian Distribution")
curve(pnig(x, param = param), from = nigRange[1], to = nigRange[2],
      n = 1000)
title("Distribution Function of the\n Normal Inverse Gaussian Distribution")
```

```

dataVector <- rnig(500, param = param)
curve(dnig(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram\n of the Normal Inverse Gaussian Distribution")
DistributionUtils::logHist(dataVector, main =
  "Log-Density and Log-Histogram\n of the Normal Inverse Gaussian Distribution")
curve(log(dnig(x, param = param)), add = TRUE,
      range(dataVector)[1], range(dataVector)[2], n = 500)
par(mfrow = c(2, 1))
curve(dnig(x, param = param), from = nigRange[1], to = nigRange[2],
      n = 1000)
title("Density of the\n Normal Inverse Gaussian Distribution")
curve(ddnig(x, param = param), from = nigRange[1], to = nigRange[2],
      n = 1000)
title("Derivative of the Density\n of the Normal Inverse Gaussian Distribution")

```

nigCalcRange

Range of a normal inverse Gaussian Distribution

Description

Given the parameter vector `param` of a normal inverse Gaussian distribution, this function calculates the range outside of which the distribution has negligible probability, or the density function is negligible, to a specified tolerance. The parameterization used is the (α, β) one (see [dnig](#)). To use another parameterization, use [hyperbChangePars](#).

Usage

```

nigCalcRange(mu = 0, delta = 1, alpha = 1, beta = 0,
             param = c(mu, delta, alpha, beta),
             tol = 10^(-5), density = TRUE, ...)

```

Arguments

<code>mu</code>	μ is the location parameter. By default this is set to 0.
<code>delta</code>	δ is the scale parameter of the distribution. A default value of 1 has been set.
<code>alpha</code>	α is the tail parameter, with a default value of 1.
<code>beta</code>	β is the skewness parameter, by default this is 0.
<code>param</code>	Value of parameter vector specifying the normal inverse Gaussian distribution. This takes the form <code>c(mu, delta, alpha, beta)</code> .
<code>tol</code>	Tolerance.
<code>density</code>	Logical. If FALSE, the bounds are for the probability distribution. If TRUE, they are for the density function.
<code>...</code>	Extra arguments for calls to uniroot .

Details

The particular normal inverse Gaussian distribution being considered is specified by the parameter value `param`.

If `density = FALSE`, the function calculates the effective range of the distribution, which is used in calculating the distribution function and quantiles, and may be used in determining the range when plotting the distribution. By effective range is meant that the probability of an observation being greater than the upper end is less than the specified tolerance `tol`. Likewise for being smaller than the lower end of the range. Note that this has not been implemented yet.

If `density = TRUE`, the function gives a range, outside of which the density is less than the given tolerance. Useful for plotting the density.

Value

A two-component vector giving the lower and upper ends of the range.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong

References

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley

See Also

[dnig](#), [hyperbChangePars](#)

Examples

```
par(mfrow = c(1, 2))
param <- c(0, 1, 3, 1)
nigRange <- nigCalcRange(param = param, tol = 10^(-3))
nigRange
curve(pnig(x, param = param), nigRange[1], nigRange[2])
maxDens <- dnig(nigMode(param = param), param = param)
nigRange <- nigCalcRange(param = param, tol = 10^(-3) * maxDens, density = TRUE)
nigRange
curve(dnig(x, param = param), nigRange[1], nigRange[2])
```

nigFit

*Fit the normal inverse Gaussian Distribution to Data***Description**

Fits a normal inverse Gaussian distribution to data. Displays the histogram, log-histogram (both with fitted densities), Q-Q plot and P-P plot for the fit which has the maximum likelihood.

Usage

```
nigFit(x, freq = NULL, paramStart = NULL,
       startMethod = c("Nelder-Mead", "BFGS"),
       startValues = c("FN", "Cauchy", "MoM", "US"),
       criterion = "MLE",
       method = c("Nelder-Mead", "BFGS", "nlm",
                 "L-BFGS-B", "nlminb", "constrOptim"),
       plots = FALSE, printOut = FALSE,
       controlBFGS = list(maxit = 200),
       controlNM = list(maxit = 1000), maxitNLM = 1500,
       controlLBFGSB = list(maxit = 200),
       controlNLMINB = list(),
       controlCO = list(), ...)

## S3 method for class 'nigFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'nigFit'
plot(x, which = 1:4,
     plotTitles = paste(c("Histogram of ", "Log-Histogram of ",
                          "Q-Q Plot of ", "P-P Plot of "), x$obsName,
                        sep = "" ),
     ask = prod(par("mfcol")) < length(which) & dev.interactive(), ...)

## S3 method for class 'nigFit'
coef(object, ...)

## S3 method for class 'nigFit'
vcov(object, ...)
```

Arguments

x	Data vector for nigFit. Object of class "nigFit" for print.nigFit and plot.nigFit.
freq	A vector of weights with length equal to length(x).
paramStart	A user specified starting parameter vector param taking the form c(mu, delta, alpha, beta).
startMethod	Method used by nigFitStart in calls to optim .

startValues	Code giving the method of determining starting values for finding the maximum likelihood estimate of param.
criterion	Currently only "MLE" is implemented.
method	Different optimisation methods to consider. See Details .
plots	Logical. If FALSE suppresses printing of the histogram, log-histogram, Q-Q plot and P-P plot.
printOut	Logical. If FALSE suppresses printing of results of fitting.
controlBFGS	A list of control parameters for <code>optim</code> when using the "BFGS" optimisation.
controlNLM	A list of control parameters for <code>optim</code> when using the "Nelder-Mead" optimisation.
maxitNLM	A positive integer specifying the maximum number of iterations when using the "nlm" optimisation.
controlLBFGSB	A list of control parameters for <code>optim</code> when using the "L-BFGS-B" optimisation.
controlNLMINB	A list of control parameters for <code>nlminb</code> when using the "nlminb" optimisation.
controlCO	A list of control parameters for <code>constrOptim</code> when using the "constrOptim" optimisation.
digits	Desired number of digits when the object is printed.
which	If a subset of the plots is required, specify a subset of the numbers 1:4.
plotTitles	Titles to appear above the plots.
ask	Logical. If TRUE, the user is <i>asked</i> before each plot, see <code>par(ask = .)</code> .
...	Passes arguments to <code>par</code> , <code>hist</code> , <code>logHist</code> , <code>qqnig</code> and <code>ppnig</code> .
object	Object of class "nigFit" for <code>coef.nigFit</code> and for <code>vcov.nigFit</code> .

Details

`startMethod` can be either "BFGS" or "Nelder-Mead".

`startValues` can be one of the following:

- "US" User-supplied.
- "FN" A fitted normal distribution.
- "Cauchy" Based on a fitted Cauchy distribution.
- "MoM" Method of moments.

For the details concerning the use of `paramStart`, `startMethod`, and `startValues`, see [nigFitStart](#).

The three optimisation methods currently available are:

- "BFGS" Uses the quasi-Newton method "BFGS" as documented in [optim](#).
- "Nelder-Mead" Uses an implementation of the Nelder and Mead method as documented in [optim](#).
- "nlm" Uses the `nlm` function in R.

For details of how to pass control information for optimisation using [optim](#) and `nlm`, see [optim](#) and `nlm`.

When `method = "nlm"` is used, warnings may be produced. These do not appear to be a problem.

Value

A list with components:

param	A vector giving the maximum likelihood estimate of param, as $c(\mu, \delta, \alpha, \beta)$.
maxLik	The value of the maximised log-likelihood.
method	Optimisation method used.
conv	Convergence code. See the relevant documentation (either optim or nlm) for details on convergence.
iter	Number of iterations of optimisation routine.
x	The data used to fit the normal inverse Gaussian distribution.
xName	A character string with the actual x argument name.
paramStart	Starting value of param returned by call to nigFitStart .
svName	Descriptive name for the method finding start values.
startValues	Acronym for the method of finding start values.
breaks	The cell boundaries found by a call to hist .
midpoints	The cell midpoints found by a call to hist .
empDens	The estimated density found by a call to hist .

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong

References

- Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.*, **A353**, 401–419.
- Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.*, **41**, 127–146.
- Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley

See Also

[optim](#), [nlm](#), [par](#), [hist](#), [logHist](#), [qqnig](#), [ppnig](#), [dskewlap](#) and [nigFitStart](#).

Examples

```
param <- c(2, 2, 2, 1)
dataVector <- rnig(500, param = param)
## See how well nigFit works
nigFit(dataVector)
nigFit(dataVector, plots = TRUE)
fit <- nigFit(dataVector)
par(mfrow = c(1, 2))
plot(fit, which = c(1, 3))

## Use nlm instead of default
nigFit(dataVector, method = "nlm")
```

nigFitStart	<i>Find Starting Values for Fitting a normal inverse Gaussian Distribution</i>
-------------	--

Description

Finds starting values for input to a maximum likelihood routine for fitting normal inverse Gaussian distribution to data.

Usage

```
nigFitStart(x, startValues = c("FN", "Cauchy", "MoM", "US"),
            paramStart = NULL,
            startMethodMoM = c("Nelder-Mead", "BFGS"), ...)
nigFitStartMoM(x, startMethodMoM = "Nelder-Mead", ...)
```

Arguments

x	data vector.
startValues	a character string specifying the method for starting values to consider. See Details .
paramStart	starting values for param if startValues = "US".
startMethodMoM	Method used by call to optim in finding method of moments estimates.
...	Passes arguments to hist and optim .

Details

Possible values of the argument startValues are the following:

- "US" User-supplied.
- "FN" A fitted normal distribution.
- "Cauchy" Based on a fitted Cauchy distribution, from [fitdistr\(\)](#) of the **MASS** package.
- "MoM" Method of moments.

If startValues = "US" then a value must be supplied for paramStart.

If startValues = "MoM", nigFitStartMoM is called. If startValues = "MoM" an initial optimisation is needed to find the starting values. These optimisations call [optim](#).

Value

nigFitStart returns a list with components:

paramStart	A vector with elements mu, delta, alpha and beta giving the starting value of param.
xName	A character string with the actual x argument name.

breaks The cell boundaries found by a call to [hist](#).
midpoints The cell midpoints found by a call to [hist](#).
empDens The estimated density found by a call to [hist](#).

nigFitStartMoM returns only the method of moments estimates as a vector with elements mu, delta, alpha and beta.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong

References

Barndorff-Nielsen, O. (1977) Exponentially decreasing distributions for the logarithm of particle size, *Proc. Roy. Soc. Lond.*, **A353**, 401–419.
Barndorff-Nielsen, O., Blæsild, P., Jensen, J., and Sörenson, M. (1985). The fascination of sand. In *A celebration of statistics, The ISI Centenary Volume*, eds., Atkinson, A. C. and Fienberg, S. E., pp. 57–87. New York: Springer-Verlag.
Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.*, **41**, 127–146.

See Also

[dnig](#), [dskewlap](#), [nigFit](#), [hist](#), [optim](#), [fitdistr](#).

Examples

```
param <- c(2, 2, 2, 1)
dataVector <- rnig(500, param = param)
nigFitStart(dataVector, startValues = "FN")
nigFitStartMoM(dataVector)
nigFitStart(dataVector, startValues = "MoM")
```

nigHessian	<i>Calculate Two-Sided Hessian for the Normal Inverse Gaussian Distribution</i>
------------	---

Description

Calculates the Hessian of a function, either exactly or approximately. Used to obtaining the information matrix for maximum likelihood estimation.

Usage

```
nigHessian(x, param, hessianMethod = "tsHessian",
           whichParam = 1:5, ...)
```


Arguments

x	Data vector.
param	The maximum likelihood estimates parameter vector of the normal inverse Gaussian distribution. The normal inverse Gaussian distribution has the same sets of parameterizations as the hyperbolic distribution. There are five different sets of parameterizations can be used in this function, the first four sets are listed in <code>hyperbChangePars</code> and the last set is the log scale of the first set of the parameterization, i.e., $\mu, \log(\delta), \Pi, \log(\zeta)$.
hessianMethod	Only the approximate method (" <code>tsHessian</code> ") has actually been implemented so far.
whichParam	Numeric. A number between 1 to 5 indicating which set of the parameterization is the specified value in argument <code>param</code> belong to.
...	Values of other parameters of the function <code>fun</code> if required.

Details

The approximate Hessian is obtained via a call to `tsHessian` from the package `DistributionUtils`. `summary.nigFit` calls the function `nigHessian` to calculate the Hessian matrix when the argument `hessian = TRUE`.

Value

`nigHessian` gives the approximate or exact Hessian matrix for the data vector `x` and the estimated parameter vector `param`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

Examples

```
### Calculate the exact Hessian using nigHessian:
param <- c(2, 2, 2, 1)
dataVector <- rnig(500, param = param)
fit <- nigFit(dataVector, method = "BFGS")
coef=coef(fit)
nigHessian(x=dataVector, param=coef, hessianMethod = "tsHessian",
           whichParam = 2)

### Or calculate the exact Hessian using summary.nigFit method:
### summary(fit, hessian = TRUE)

## Calculate the approximate Hessian:
summary(fit, hessian = TRUE, hessianMethod = "tsHessian")
```

Description

These objects store different parameter sets of the normal inverse Gaussian distribution as matrices for testing or demonstration purposes.

The parameter sets `nigSmallShape` and `nigLargeShape` have a constant location parameter of $\mu = 0$, and constant scale parameter $\delta = 1$. In `nigSmallParam` and `nigLargeParam` the values of the location and scale parameters vary. In these parameter sets the location parameter $\mu = 0$ takes values from $\{0, 1\}$ and $\{-1, 0, 1, 2\}$ respectively. For the scale parameter δ , values are drawn from $\{1, 5\}$ and $\{1, 2, 5, 10\}$ respectively.

For the shape parameters α and β the approach is more complex. The values for these shape parameters were chosen by choosing values of ξ and χ which range over the shape triangle, then the function `nigChangePars` was applied to convert them to the α, β parameterization. The resulting α, β values were then rounded to three decimal places. See the examples for the values of ξ and χ for the large parameter sets.

Usage

```
nigSmallShape
nigLargeShape
nigSmallParam
nigLargeParam
```

Format

`nigSmallShape`: a 7 by 4 matrix; `nigLargeShape`: a 15 by 4 matrix; `nigSmallParam`: a 28 by 4 matrix; `nigLargeParam`: a 240 by 4 matrix.

Author(s)

David Scott <d.scott@auckland.ac.nz>

Examples

```
data(nigParam)
plotShapeTriangle()
xis <- rep(c(0.1,0.3,0.5,0.7,0.9), 1:5)
chis <- c(0,-0.25,0.25,-0.45,0,0.45,-0.65,-0.3,0.3,0.65,
         -0.85,-0.4,0,0.4,0.85)
points(chis, xis, pch = 20, col = "red")

## Testing the accuracy of nigMean
for (i in 1:nrow(nigSmallParam)) {
  param <- nigSmallParam[i, ]
  x <- rnig(1000, param = param)
```

```

sampleMean <- mean(x)
funMean <- nigMean(param = param)
difference <- abs(sampleMean - funMean)
print(difference)
}

```

nigPlots	<i>Normal inverse Gaussian Quantile-Quantile and Percent-Percent Plots</i>
----------	--

Description

qqnig produces a normal inverse Gaussian Q-Q plot of the values in y .

ppnig produces a normal inverse Gaussian P-P (percent-percent) or probability plot of the values in y .

Graphical parameters may be given as arguments to qqnig, and ppnig.

Usage

```

qqnig(y, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta),
      main = "Normal inverse Gaussian Q-Q Plot",
      xlab = "Theoretical Quantiles",
      ylab = "Sample Quantiles",
      plot.it = TRUE, line = TRUE, ...)

```

```

ppnig(y, mu = 0, delta = 1, alpha = 1, beta = 0,
      param = c(mu, delta, alpha, beta),
      main = "Normal inverse Gaussian P-P Plot",
      xlab = "Uniform Quantiles",
      ylab = "Probability-integral-transformed Data",
      plot.it = TRUE, line = TRUE, ...)

```

Arguments

y	The data sample.
μ	μ is the location parameter. By default this is set to 0.
δ	δ is the scale parameter of the distribution. A default value of 1 has been set.
α	α is the tail parameter, with a default value of 1.
β	β is the skewness parameter, by default this is 0.
param	Parameters of the normal inverse Gaussian distribution.
xlab, ylab, main	Plot labels.
plot.it	Logical. Should the result be plotted?
line	Add line through origin with unit slope.
...	Further graphical parameters.

Value

For qqnig and ppnig, a list with components:

x	The x coordinates of the points that are to be plotted.
y	The y coordinates of the points that are to be plotted.

References

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

See Also

[ppoints](#), [dnig](#), [nigFit](#)

Examples

```
par(mfrow = c(1, 2))
param <- c(2, 2, 2, 1.5)
y <- rnig(200, param = param)
qqnig(y, param = param, line = FALSE)
abline(0, 1, col = 2)
ppnig(y, param = param)
```

plotShapeTriangle	<i>Plot the Shape Triangle</i>
-------------------	--------------------------------

Description

Plots the shape triangle for a hyperbolic distribution or generalized hyperbolic distribution. For the hyperbolic distribution the parameter χ is related to the skewness, and the parameter ξ is related to the kurtosis. See Barndorff-Nielsen, O. and Blæsild, P. (1981).

Usage

```
plotShapeTriangle(xgap = 0.025, ygap = 0.0625/2,
                 main = "Shape Triangle", ...)
```

Arguments

xgap	Gap between the left- and right-hand edges of the shape triangle and the border surrounding the graph.
ygap	Gap between the top and bottom of the shape triangle and the border surrounding the graph.
main	Title for the plot.
...	Values of other graphical parameters.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Barndorff-Nielsen, O. and Blæsild, P (1981). Hyperbolic distributions and ramifications: contributions to theory and application. In *Statistical Distributions in Scientific Work*, eds., Taillie, C., Patil, G. P., and Baldessari, B. A., Vol. 4, pp. 19–44. Dordrecht: Reidel.

Examples

```
plotShapeTriangle()
```

resistors

Resistance of One-half-ohm Resistors

Description

This data set gives the resistance in ohms of 500 nominally one-half-ohm resistors, presented in Hahn and Shapiro (1967). Summary data giving the frequency of observations in 28 intervals.

Usage

```
data(resistors)
```

Format

The resistors data frame has 28 rows and 2 columns.

[, 1]	midpoints	midpoints of intervals (ohm)
[, 2]	counts	number of observations in interval

Source

Hahn, Gerald J. and Shapiro, Samuel S. (1967) *Statistical Models in Engineering*. New York: Wiley, page 207.

References

Chen, Hanfeng, and Kamburowska, Grazyna (2001) Fitting data to the Johnson system. *J. Statist. Comput. Simul.* **70**, 21–32.

Examples

```
data(resistors)
str(resistors)
### Construct data from frequency summary, taking all observations
```

```

### at midpoints of intervals
resistances <- rep(resistors$midpoints, resistors$counts)
hist(resistances)
DistributionUtils::logHist(resistances)
## Fit the hyperbolic distribution
hyperbFit(resistances)

## Actually fit.hyperb can deal with frequency data
hyperbFit(resistors$midpoints, freq = resistors$counts)

```

SandP500

S&P 500

Description

This data set gives the value of Standard and Poor's most notable stock market price index (the S&P 500) at year end, from 1800 to 2001.

Usage

```
data(SandP500)
```

Format

A vector of 202 observations.

Source

At the time, <http://www.globalfindata.com> which no longer exists.

References

Brown, Barry W., Spears, Floyd M. and Levy, Lawrence B. (2002) The log F : a distribution for all seasons. *Computational Statistics*, **17**, 47–58.

Examples

```

data(SandP500)
### Consider proportional changes in the index
change <- SandP500[-length(SandP500)] / SandP500[-1]
hist(change)
### Fit hyperbolic distribution to changes
hyperbFit(change)

```

SkewLaplace

*Skew-Laplace Distribution***Description**

Density function, distribution function, quantiles and random number generation for the skew-Laplace distribution.

Usage

```

dskewlap(x, mu = 0, alpha = 1, beta = 1,
          param = c(mu, alpha, beta), logPars = FALSE)
pskewlap(q, mu = 0, alpha = 1, beta = 1,
          param = c(mu, alpha, beta))
qskewlap(p, mu = 0, alpha = 1, beta = 1,
          param = c(mu, alpha, beta))
rskewlap(n, mu = 0, alpha = 1, beta = 1,
          param = c(mu, alpha, beta))

```

Arguments

x, q	Vector of quantiles.
p	Vector of probabilities.
n	Number of observations to be generated.
mu	The location parameter, set to 0 by default.
alpha, beta	The shape parameters, both set to 1 by default.
param	Vector of parameters of the skew-Laplace distribution: μ , α and β .
logPars	Logical. If TRUE the second and third components of param are taken to be $\log(\alpha)$ and $\log(\beta)$ respectively.

Details

The central skew-Laplace has mode zero, and is a mixture of a (negative) exponential distribution with mean β , and the negative of an exponential distribution with mean α . The weights of the positive and negative components are proportional to their means.

The general skew-Laplace distribution is a shifted central skew-Laplace distribution, where the mode is given by μ .

The density is given by:

$$f(x) = \frac{1}{\alpha + \beta} e^{(x-\mu)/\alpha}$$

for $x \leq \mu$, and

$$f(x) = \frac{1}{\alpha + \beta} e^{-(x-\mu)/\beta}$$

for $x \geq \mu$

Value

dskewlap gives the density, pskewlap gives the distribution function, qskewlap gives the quantile function and rskewlap generates random variates. The distribution function is obtained by elementary integration of the density function. Random variates are generated from exponential observations using the characterization of the skew-Laplace as a mixture of exponential observations.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Ai-Wei Lee, Richard Trendall

References

Fieller, N. J., Flenley, E. C. and Olbricht, W. (1992) Statistics of particle size data. *Appl. Statist.*, **41**, 127–146.

See Also

[hyperbFitStart](#)

Examples

```
param <- c(1, 1, 2)
par(mfrow = c(1, 2))
curve(dskewlap(x, param = param), from = -5, to = 8, n = 1000)
title("Density of the\n Skew-Laplace Distribution")
curve(pskewlap(x, param = param), from = -5, to = 8, n = 1000)
title("Distribution Function of the\n Skew-Laplace Distribution")
dataVector <- rskewlap(500, param = param)
curve(dskewlap(x, param = param), range(dataVector)[1], range(dataVector)[2],
      n = 500)
hist(dataVector, freq = FALSE, add = TRUE)
title("Density and Histogram\n of the Skew-Laplace Distribution")
DistributionUtils::logHist(dataVector, main =
  "Log-Density and Log-Histogram\n of the Skew-Laplace Distribution")
curve(log(dskewlap(x, param = param)), add = TRUE,
      range(dataVector)[1], range(dataVector)[2], n = 500)
```

SkewLaplacePlots

Skew-Laplace Quantile-Quantile and Percent-Percent Plots

Description

qqskewlap produces a skew-Laplace QQ plot of the values in y.

ppskewlap produces a skew-Laplace PP (percent-percent) or probability plot of the values in y.

If line = TRUE, a line with zero intercept and unit slope is added to the plot.

Graphical parameters may be given as arguments to qqskewLap, and ppskewLap.

Usage

```

qqskewlap(y, mu = 0, alpha = 1, beta = 1,
           param = c(mu, alpha, beta),
           main = "Skew-Laplace Q-Q Plot",
           xlab = "Theoretical Quantiles",
           ylab = "Sample Quantiles",
           plot.it = TRUE, line = TRUE, ...)

ppskewlap(y, mu = 0, alpha = 1, beta = 1,
           param = c(mu, alpha, beta),
           main = "Skew-Laplace P-P Plot",
           xlab = "Uniform Quantiles",
           ylab = "Probability-integral-transformed Data",
           plot.it = TRUE, line = TRUE, ...)

```

Arguments

<code>y</code>	The data sample.
<code>mu</code>	The location parameter, set to 0 by default.
<code>alpha, beta</code>	The shape parameters, both set to 1 by default.
<code>param</code>	Parameters of the skew-Laplace distribution.
<code>xlab, ylab, main</code>	Plot labels.
<code>plot.it</code>	Logical. TRUE denotes the results should be plotted.
<code>line</code>	Logical. If TRUE, a line with zero intercept and unit slope is added to the plot.
<code>...</code>	Further graphical parameters.

Value

For `qqskewlap` and `ppskewlap`, a list with components:

<code>x</code>	The x coordinates of the points that are be plotted.
<code>y</code>	The y coordinates of the points that are be plotted.

References

Wilk, M. B. and Gnanadesikan, R. (1968) Probability plotting methods for the analysis of data. *Biometrika*. **55**, 1–17.

See Also

[ppoints](#), [dskewlap](#).

Examples

```

par(mfrow = c(1, 2))
y <- rskewlap(1000, param = c(2, 0.5, 1))
qqskewlap(y, param = c(2, 0.5, 1), line = FALSE)
abline(0, 1, col = 2)
ppskewlap(y, param = c(2, 0.5, 1))

```

Specific Generalized Hyperbolic Moments and Mode

Moments and Mode of the Generalized Hyperbolic Distribution

Description

Functions to calculate the mean, variance, skewness, kurtosis and mode of a specific generalized hyperbolic distribution.

Usage

```

ghypMean(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
         param = c(mu, delta, alpha, beta, lambda))
ghypVar(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
        param = c(mu, delta, alpha, beta, lambda))
ghypSkew(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
         param = c(mu, delta, alpha, beta, lambda))
ghypKurt(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
         param = c(mu, delta, alpha, beta, lambda))
ghypMode(mu = 0, delta = 1, alpha = 1, beta = 0, lambda = 1,
         param = c(mu, delta, alpha, beta, lambda))

```

Arguments

mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
lambda	λ is the shape parameter and dictates the shape that the distribution shall take. Default value is 1.
param	Parameter vector of the generalized hyperbolic distribution.

Value

ghypMean gives the mean of the generalized hyperbolic distribution, ghypVar the variance, ghypSkew the skewness, ghypKurt the kurtosis, and ghypMode the mode. The formulae used for the mean is given in Prause (1999). The variance, skewness and kurtosis are obtained using the recursive formula implemented in [ghypMom](#) which can calculate moments of all orders about any point.

The mode is found by a numerical optimisation using `optim`. For the special case of the hyperbolic distribution a formula for the mode is available, see [hyperbMode](#).

The parameterization of the generalized hyperbolic distribution used for these functions is the (α, β) one. See [ghypChangePars](#) to transfer between parameterizations.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Thomas Tran

References

Prause, K. (1999) *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.

See Also

[dghyp](#), [ghypChangePars](#), [besselK](#), [RLambda](#).

Examples

```
param <- c(2, 2, 2, 1, 2)
ghypMean(param = param)
ghypVar(param = param)
ghypSkew(param = param)
ghypKurt(param = param)
ghypMode(param = param)
maxDens <- dghyp(ghypMode(param = param), param = param)
ghypRange <- ghypCalcRange(param = param, tol = 10-3 * maxDens)
curve(dghyp(x, param = param), ghypRange[1], ghypRange[2])
abline(v = ghypMode(param = param), col = "blue")
abline(v = ghypMean(param = param), col = "red")
```

Specific Generalized Inverse Gaussian Moments and Mode

Moments and Mode of the Generalized Inverse Gaussian Distribution

Description

Functions to calculate the mean, variance, skewness, kurtosis and mode of a specific generalized inverse Gaussian distribution.

Usage

```
gigMean(chi = 1, psi = 1, lambda = 1,
        param = c(chi, psi, lambda))
gigVar(chi = 1, psi = 1, lambda = 1,
       param = c(chi, psi, lambda))
gigSkew(chi = 1, psi = 1, lambda = 1,
```

```

      param = c(chi, psi, lambda))
gigKurt(chi = 1, psi = 1, lambda = 1,
      param = c(chi, psi, lambda))
gigMode(chi = 1, psi = 1, lambda = 1,
      param = c(chi, psi, lambda))

```

Arguments

chi	A shape parameter that by default holds a value of 1.
psi	Another shape parameter that is set to 1 by default.
lambda	Shape parameter of the GIG distribution. Common to all forms of parameterization. By default this is set to 1.
param	Parameter vector of the generalized inverse Gaussian distribution.

Value

gigMean gives the mean of the generalized inverse Gaussian distribution, gigVar the variance, gigSkew the skewness, gigKurt the kurtosis, and gigMode the mode. The formulae used are as given in Jorgensen (1982), pp. 13–17. Note that the kurtosis is the standardised fourth cumulant or what is sometimes called the kurtosis excess. (See <http://mathworld.wolfram.com/Kurtosis.html> for a discussion.)

The parameterization used for the generalized inverse Gaussian distribution is the (χ, ψ) one (see [dgig](#)). To use another parameterization, use [gigChangePars](#).

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

Jorgensen, B. (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution*. Lecture Notes in Statistics, Vol. 9, Springer-Verlag, New York.

See Also

[dgig](#), [gigChangePars](#), [besselK](#)

Examples

```

param <- c(5, 2.5, -0.5)
gigMean(param = param)
gigVar(param = param)
gigSkew(param = param)
gigKurt(param = param)
gigMode(param = param)

```

Specific Hyperbolic Distribution Moments and Mode
Moments and Mode of the Hyperbolic Distribution

Description

Functions to calculate the mean, variance, skewness, kurtosis and mode of a specific hyperbolic distribution.

Usage

```
hyperbMean(mu = 0, delta = 1, alpha = 1, beta = 0,
           param = c(mu, delta, alpha, beta))
hyperbVar(mu = 0, delta = 1, alpha = 1, beta = 0,
          param = c(mu, delta, alpha, beta))
hyperbSkew(mu = 0, delta = 1, alpha = 1, beta = 0,
           param = c(mu, delta, alpha, beta))
hyperbKurt(mu = 0, delta = 1, alpha = 1, beta = 0,
           param = c(mu, delta, alpha, beta))
hyperbMode(mu = 0, delta = 1, alpha = 1, beta = 0,
           param = c(mu, delta, alpha, beta))
```

Arguments

mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
param	Parameter vector of the hyperbolic distribution.

Details

The formulae used for the mean, variance and mode are as given in Barndorff-Nielsen and Blæsild (1983), p. 702. The formulae used for the skewness and kurtosis are those of Barndorff-Nielsen and Blæsild (1981), Appendix 2.

Note that the variance, skewness and kurtosis can be obtained from the functions for the generalized hyperbolic distribution as special cases. Likewise other moments can be obtained from the function [ghypMom](#) which implements a recursive method to moments of any desired order. Note that functions for the generalized hyperbolic distribution use a different parameterization, so care is required.

Value

hyperbMean gives the mean of the hyperbolic distribution, hyperbVar the variance, hyperbSkew the skewness, hyperbKurt the kurtosis and hyperbMode the mode.

Note that the kurtosis is the standardised fourth cumulant or what is sometimes called the kurtosis excess. (See <http://mathworld.wolfram.com/Kurtosis.html> for a discussion.)

The parameterization of the hyperbolic distribution used for this and other components of the GeneralizedHyperbolic package is the (α, β) one. See [hyperbChangePars](#) to transfer between parameterizations.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Richard Trendall, Thomas Tran

References

Barndorff-Nielsen, O. and Blæsild, P (1981). Hyperbolic distributions and ramifications: contributions to theory and application. In *Statistical Distributions in Scientific Work*, eds., Taillie, C., Patil, G. P., and Baldessari, B. A., Vol. 4, pp. 19–44. Dordrecht: Reidel.

Barndorff-Nielsen, O. and Blæsild, P (1983). Hyperbolic distributions. In *Encyclopedia of Statistical Sciences*, eds., Johnson, N. L., Kotz, S. and Read, C. B., Vol. 3, pp. 700–707. New York: Wiley.

See Also

[dhyperb](#), [hyperbChangePars](#), [besselK](#), [ghypMom](#), [ghypMean](#), [ghypVar](#), [ghypSkew](#), [ghypKurt](#)

Examples

```
param <- c(2, 2, 2, 1)
hyperbMean(param = param)
hyperbVar(param = param)
hyperbSkew(param = param)
hyperbKurt(param = param)
hyperbMode(param = param)
```

Specific Normal Inverse Gaussian Distribution Moments and Mode

Moments and Mode of the Normal Inverse Gaussian Distribution

Description

Functions to calculate the mean, variance, skewness, kurtosis and mode of a specific normal inverse Gaussian distribution.

Usage

```
nigMean(mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))
nigVar(mu = 0, delta = 1, alpha = 1, beta = 0,
       param = c(mu, delta, alpha, beta))
nigSkew(mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))
nigKurt(mu = 0, delta = 1, alpha = 1, beta = 0,
```

```

    param = c(mu, delta, alpha, beta))
nigMode(mu = 0, delta = 1, alpha = 1, beta = 0,
        param = c(mu, delta, alpha, beta))

```

Arguments

mu	μ is the location parameter. By default this is set to 0.
delta	δ is the scale parameter of the distribution. A default value of 1 has been set.
alpha	α is the tail parameter, with a default value of 1.
beta	β is the skewness parameter, by default this is 0.
param	Parameter vector of the normal inverse Gaussian distribution.

Details

The mean, variance, skewness, kurtosis and mode for the normal inverse Gaussian distribution can be obtained from the functions for the generalized hyperbolic distribution as special cases (i.e., $\lambda = -1/2$). Likewise other moments can be obtained from the function [ghypMom](#) which implements a recursive method to moments of any desired order.

The proper formulae for the mean, variance and skewness of the normal inverse Gaussian distribution can be found in Paolella, Marc S. (2007), Chapter 9, p325.

Value

`nigMean` gives the mean of the normal inverse Gaussian distribution, `nigVar` the variance, `nigSkew` the skewness, `nigKurt` the kurtosis and `nigMode` the mode.

Note that the kurtosis is the standardised fourth cumulant or what is sometimes called the kurtosis excess. (See <http://mathworld.wolfram.com/Kurtosis.html> for a discussion.)

The parameterization of the normal inverse Gaussian distribution used for this and other components of the `GeneralizedHyperbolic` package is the (α, β) one. See [hyperbChangePars](#) to transfer between parameterizations.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong

References

Paolella, Marc S. (2007) *Intermediate Probability: A Computational Approach*, Chichester: Wiley

See Also

[dnig](#), [hyperbChangePars](#), [besselK](#), [ghypMom](#), [ghypMean](#), [ghypVar](#), [ghypSkew](#), [ghypKurt](#)

Examples

```
param <- c(2, 2, 2, 1)
nigMean(param = param)
nigVar(param = param)
nigSkew(param = param)
nigKurt(param = param)
nigMode(param = param)
```

summary.gigFit

*Summarizing Normal Inverse Gaussian Distribution Fit***Description**

summary Method for class "gigFit".

Usage

```
## S3 method for class 'gigFit'
summary(object, hessian = FALSE,
        hessianMethod = "tsHessian", ...)

## S3 method for class 'summary.gigFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "gigFit", resulting from a call to gigFit .
hessian	Logical. If TRUE the Hessian is printed.
hessianMethod	The two-sided Hessian approximation given by <code>tsHessian</code> from the package <code>DistributionUtils</code> is the only method implemented so far.
x	An object of class "summary.gigFit", resulting from a call to <code>summary.gigFit</code> .
digits	The number of significant digits to use when printing.
...	Further arguments passed to or from other methods.

Details

If `hessian = FALSE` no calculations are performed, the class of object is simply changed from `gigFit` to `summary.gigFit` so that it can be passed to `print.summary.gigFit` for printing in a convenient form.

If `hessian = TRUE` the Hessian is calculated via a call to [gigHessian](#) and the standard errors of the parameter estimates are calculated using the Hessian and these are added to the original list object. The class of the object returned is again changed to `summary.gigFit`.

Value

summary.gigFit returns a list comprised of the original object object and additional elements hessian and sds if hessian = TRUE, otherwise it returns the original object. The class of the object returned is changed to summary.gigFit.

See [gigFit](#) for the composition of an object of class gigFit.

If the Hessian and standard errors have not been added to the object x, print.summary.gigFit prints a summary in the same format as [print.gigFit](#). When the Hessian and standard errors are available, the Hessian is printed and the standard errors for the parameter estimates are printed in parentheses beneath the parameter estimates, in the manner of fitdistr in the package MASS.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

See Also

[gigFit](#), [summary](#), [gigHessian](#).

Examples

```
### Continuing the gigFit(.) example:
param <- c(1,1,1)
dataVector <- rgig(500, param = param)
fit <- gigFit(dataVector)
print(fit)
summary(fit, hessian = TRUE, hessianMethod = "tsHessian")
```

summary.hyperbFit	<i>Summarizing Hyperbolic Distribution Fit</i>
-------------------	--

Description

summary Method for class "hyperbFit".

Usage

```
## S3 method for class 'hyperbFit'
summary(object, hessian = FALSE,
         hessianMethod = "exact", ...)

## S3 method for class 'summary.hyperbFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "hyperbFit", resulting from a call to hyperbFit .
hessian	Logical. If TRUE the Hessian is printed.
hessianMethod	Two methods are available to calculate the Hessian exactly ("exact") or approximately ("tsHessian").
x	An object of class "summary.hyperbFit", resulting from a call to <code>summary.hyperbFit</code> .
digits	The number of significant digits to use when printing.
...	Further arguments passed to or from other methods.

Details

If `hessian = FALSE` no calculations are performed, the class of object is simply changed from `hyperbFit` to `summary.hyperbFit` so that it can be passed to `print.summary.hyperbFit` for printing in a convenient form.

If `hessian = TRUE` the Hessian is calculated via a call to [hyperbHessian](#) and the standard errors of the parameter estimates are calculated using the Hessian and these are added to the original list object. The class of the object returned is again changed to `summary.hyperbFit`.

Value

`summary.hyperbFit` returns a list comprised of the original object `object` and additional elements `hessian` and `sds` if `hessian = TRUE`, otherwise it returns the original object. The class of the object returned is changed to `summary.hyperbFit`.

See [hyperbFit](#) for the composition of an object of class `hyperbFit`.

If the Hessian and standard errors have not been added to the object `x`, `print.summary.hyperbFit` prints a summary in the same format as [print.hyperbFit](#). When the Hessian and standard errors are available, the Hessian is printed and the standard errors for the parameter estimates are printed in parentheses beneath the parameter estimates, in the manner of `fitdistr` in the package MASS.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

See Also

[hyperbFit](#), [summary](#), [hyperbHessian](#), [tsHessian](#).

Examples

```
### Continuing the hyperbFit(.) example:
param <- c(2, 2, 2, 1)
dataVector <- rhyperb(500, param = param)
fit <- hyperbFit(dataVector, method = "BFGS")
print(fit)
summary(fit, hessian = TRUE)
```

summary.hyperblm *Summary Output of Hyperbolic Regression*

Description

It obtains summary output from class 'hyperblm' object. The summary output includes the standard error, t-statistics, p values of the coefficients estimates. Also the estimated parameters of hyperbolic error distribution, the maximum likelihood, the stage one optimization method, the two-stage alternating iterations and the convergence code.

Usage

```
## S3 method for class 'hyperblm'
summary(object, hessian = FALSE,
        nboots = 1000, ...)

## S3 method for class 'summary.hyperblm'
print(x,
      digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "hyperblm".
x	An object of class "summary.hyperblm" resulting from a call to summary.hyperblm.
hessian	Logical. If is TRUE, the standard error is calculated by the hessian matrix and the also hessian matrix is returned. Otherwise, the standard error is approximated by bootstrapping. See Details .
nboots	Numeric. Number of bootstrap simulations to obtain the bootstrap estimate of parameters standard errors.
digits	Numeric. Desired number of digits when the object is printed.
...	Passes additional arguments to functions bSE, hyperblmhessian.

Details

The function summary.hyperblm provides two approaches to obtain the standard error of parameters due to the fact that approximated hessian matrix is not stable for such complex optimization. The first approach is by approximated hessian matrix. The setting in the argument list is hessian = TRUE. The Hessian matrix is approximated by function `tsHessian`. However it may not be reliable for some error distribution parameters, for instance, the function obtains negative variance from the Hessian matrix. The second approach is by parametric bootstrapping. The setting in the argument list is hessian = FALSE which is also the default setting. The default number of bootstrap stimulations is 1000, but users can increase this when accuracy has priority over efficiency. Although the bootstrapping is fairly slow, it provides reliable standard errors.

Value

summary.hyperblm returns an object of class summary.hyperblm which is a list containing:

coefficients	A names vector of regression coefficients.
distributionParams	A named vector of fitted hyperbolic error distribution parameters.
fitted.values	The fitted mean values.
residuals	The remaining after subtract fitted values from response.
MLE	The maximum likelihood value of the model.
method	The optimization method for stage one.
paramStart	The start values of parameters that the user specified (only where relevant).
residsParamStart	The start values of parameters returned by hyperbFitStand (only where relevant).
call	The matched call.
terms	The terms object used.
contrasts	The contrasts used (only where relevant).
xlevels	The levels of the factors used in the fitting (only where relevant).
offset	The offset used (only where relevant).
xNames	The names of each explanatory variables. If explanatory variables don't have names then they shall be named x.
yVec	The response vector.
xMatrix	The explanatory variables matrix.
iterations	Number of two-stage alternating iterations to convergency.
convergence	The convergence code for two-stage optimization: 0 if the system converged; 1 if first stage did not converge, 2 if the second stage did not converge, 3 if the both stages did not converge.
breaks	The cell boundaries found by a call the hist .
hessian	Hessian Matrix. Only where Hessian = TRUE.
tval	<i>t</i> -statistics of regression coefficient estimates.
rdf	Degrees of freedom.
pval	P-values of regression coefficients estimates.
sds	Standard errors of regression coefficient estimates.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Xinxing Li <xli053@aucklanduni.ac.nz>

References

- Barndorff-Nielsen, O. (1977). Exponentially Decreasing Distribution for the Logarithm of Particle Size. In *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Vol. 353, pp. 401–419.
- Prause, K. (1999). *The generalized hyperbolic models: Estimation, financial derivatives and risk measurement*. PhD Thesis, Mathematics Faculty, University of Freiburg.
- Trendall, Richard (2005). *hypReg: A Function for Fitting a Linear Regression Model in R with Hyperbolic Error*. Masters Thesis, Statistics Faculty, University of Auckland.
- Paolella, Marc S. (2007). *Intermediate Probability: A Computational Approach*. pp. 415 -Chichester: Wiley.
- Scott, David J. and Wurtz, Diethelm and Chalabi, Yohan, (2011). *Fitting the Hyperbolic Distribution with R: A Case Study of Optimization Techniques*. In preparation.
- Stryhn, H. and Christensen, J. (2003). *Confidence intervals by the profile likelihood method, with applications in veterinary epidemiology*. ISVEE X.

See Also

[print.summary.hyperblm](#) prints the summary output in a table. [hyperblm](#) fits linear model with hyperbolic error distribution. [print.hyperblm](#) prints the regression result in a table. [coef.hyperblm](#) obtains the regression coefficients and error distribution parameters of the fitted model. [plot.hyperblm](#) obtains a residual vs fitted value plot, a histogram of residuals with error distribution density curve on top, a histogram of log residuals with error distribution error density curve on top and a QQ plot. [tsHessian](#)

Examples

```
## stackloss data example

# airflow <- stackloss[, 1]
# temperature <- stackloss[, 2]
# acid <- stackloss[, 3]
# stack <- stackloss[, 4]

# hyperblm.fit <- hyperblm(stack ~ airflow + temperature + acid,
#                           tolerance = 1e-11)

# coef.hyperblm(hyperblm.fit)
# plot.hyperblm(hyperblm.fit, breaks = 20)
# summary.hyperblm(hyperblm.fit, hessian = FALSE)
```

summary.nigFit

Summarizing Normal Inverse Gaussian Distribution Fit

Description

summary Method for class "nigFit".

Usage

```
## S3 method for class 'nigFit'
summary(object, hessian = FALSE,
        hessianMethod = "tsHessian", ...)

## S3 method for class 'summary.nigFit'
print(x,
      digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "nigFit", resulting from a call to nigFit .
hessian	Logical. If TRUE the Hessian is printed.
hessianMethod	The two-sided Hessian approximation given by <code>tsHessian</code> from the package <code>DistributionUtils</code> is the only method implemented so far.
x	An object of class "summary.nigFit", resulting from a call to <code>summary.nigFit</code> .
digits	The number of significant digits to use when printing.
...	Further arguments passed to or from other methods.

Details

If `hessian = FALSE` no calculations are performed, the class of `object` is simply changed from `nigFit` to `summary.nigFit` so that it can be passed to `print.summary.nigFit` for printing in a convenient form.

If `hessian = TRUE` the Hessian is calculated via a call to [nigHessian](#) and the standard errors of the parameter estimates are calculated using the Hessian and these are added to the original list `object`. The class of the object returned is again changed to `summary.nigFit`.

Value

`summary.nigFit` returns a list comprised of the original object `object` and additional elements `hessian` and `sds` if `hessian = TRUE`, otherwise it returns the original object. The class of the object returned is changed to `summary.nigFit`.

See [nigFit](#) for the composition of an object of class `nigFit`.

If the Hessian and standard errors have not been added to the object `x`, `print.summary.nigFit` prints a summary in the same format as [print.nigFit](#). When the Hessian and standard errors are available, the Hessian is printed and the standard errors for the parameter estimates are printed in parentheses beneath the parameter estimates, in the manner of `fitdistr` in the package `MASS`.

Author(s)

David Scott <d.scott@auckland.ac.nz>, Christine Yang Dong <c.dong@auckland.ac.nz>

See Also

[nigFit](#), [summary](#), [nigHessian](#).

Examples

```
### Continuing the nigFit(.) example:
param <- c(2, 2, 2, 1)
dataVector <- rnig(500, param = param)
fit <- nigFit(dataVector, method = "BFGS")
print(fit)
summary(fit, hessian = TRUE, hessianMethod = "tsHessian")
```

traffic

Intervals Between Vehicles on a Road

Description

Intervals between the times that 129 successive vehicles pass a point on a road, measured in seconds.

Usage

```
data(traffic)
```

Format

The traffic data is a vector of 128 observations.

Source

Bartlett, M.S. (1963) Statistical estimation of density functions *Sankhya: The Indian Journal of Statistics*, **Series A**, Vol. **25**, No. **3**, 245–254.

Jørgensen, B. (1982) Statistical Properties of the Generalized Inverse Gaussian Distribution. *Lecture Notes in Statistics*, Vol. **9**, Springer-Verlag, New York

Examples

```
data(traffic)
str(traffic)

### Fit the generalized inverse Gaussian distribution
gigFit(traffic)
```

Index

*Topic **datasets**

- ArkansasRiver, 3
- hyperbWSqTable, 59
- manquam, 60
- nervePulse, 62
- resistors, 77
- SandP500, 78
- traffic, 95

*Topic **distribution**

- Functions for Moments, 4
- Generalized Inverse Gaussian, 5
- GeneralizedHyperbolic, 8
- GeneralizedHyperbolicDistribution, 10
- GeneralizedHyperbolicPlots, 13
- ghypCalcRange, 15
- ghypChangePars, 16
- ghypCheckPars, 18
- ghypMom, 19
- ghypScale, 22
- gigCalcRange, 23
- gigChangePars, 25
- gigCheckPars, 26
- gigFit, 27
- gigFitStart, 30
- gigMom, 33
- GIGPlots, 37
- hyperbCalcRange, 38
- hyperbChangePars, 40
- hyperbFit, 43
- hyperbFitStart, 46
- Hyperbolic, 54
- HyperbPlots, 58
- momRecursion, 61
- NIG, 63
- nigCalcRange, 66
- nigFit, 68
- nigFitStart, 71
- nigPlots, 75

- plotShapeTriangle, 76
- SkewLaplace, 79
- SkewLaplacePlots, 80
- Specific Generalized Hyperbolic Moments and Mode, 82
- Specific Generalized Inverse Gaussian Moments and Mode, 83
- Specific Hyperbolic Distribution Moments and Mode, 85
- Specific Normal Inverse Gaussian Distribution Moments and Mode, 86
- summary.gigFit, 88
- summary.hyperbFit, 89
- summary.nigFit, 93

*Topic **hplot**

- GeneralizedHyperbolicPlots, 13
- GIGPlots, 37
- HyperbPlots, 58
- nigPlots, 75
- SkewLaplacePlots, 80

*Topic **htest**

- hyperbCvMTest, 41

*Topic **print**

- hyperbCvMTest, 41

- ArkansasRiver, 3
- as.data.frame, 50

- besselK, 5, 83, 84, 86, 87

- character, 71
- coef.gigFit (gigFit), 27
- coef.hyperbFit (hyperbFit), 43
- coef.hyperblm, 53, 93
- coef.hyperblm (hyperblm), 49
- coef.nigFit (nigFit), 68
- constrOptim, 45, 46, 51, 53

- ddghyp

- (GeneralizedHyperbolicDistribution), 10
- ddgig (Generalized Inverse Gaussian), 5
- ddhyperb (Hyperbolic), 54
- ddnig (NIG), 63
- dghyp, 8, 12, 14–19, 55, 56, 65, 83
- dghyp (GeneralizedHyperbolicDistribution), 10
- dgig, 12, 23, 24, 26, 27, 32, 33, 38, 84
- dgig (Generalized Inverse Gaussian), 5
- dhyperb, 5, 12, 38, 39, 41, 47, 59, 86
- dhyperb (Hyperbolic), 54
- dnig, 66, 67, 72, 76, 87
- dnig (NIG), 63
- dskewlap, 46, 47, 70, 72, 81
- dskewlap (SkewLaplace), 79
- fitdistr, 71, 72
- formula, 50, 51
- Functions for Moments, 4
- gammaLambda1 (Functions for Moments), 4
- gammaLambda2 (Functions for Moments), 4
- gammaRawMom (gigMom), 33
- Generalized Inverse Gaussian, 5
- GeneralizedHyperbolic, 8
- GeneralizedHyperbolic-package (GeneralizedHyperbolic), 8
- GeneralizedHyperbolicDistribution, 10
- GeneralizedHyperbolicPlots, 13
- ghypCalcRange, 15
- ghypChangePars, 12, 15, 16, 16, 20, 83
- ghypCheckPars, 18
- ghypKurt, 20, 86, 87
- ghypKurt (Specific Generalized Hyperbolic Moments and Mode), 82
- ghypLargeParam (ghypParam), 21
- ghypLargeShape (ghypParam), 21
- ghypMean, 20, 86, 87
- ghypMean (Specific Generalized Hyperbolic Moments and Mode), 82
- ghypMode (Specific Generalized Hyperbolic Moments and Mode), 82
- ghypMom, 19, 82, 85–87
- ghypParam, 21
- ghypScale, 22
- ghypSkew, 20, 86, 87
- ghypSkew (Specific Generalized Hyperbolic Moments and Mode), 82
- ghypSmallParam (ghypParam), 21
- ghypSmallShape (ghypParam), 21
- ghypVar, 20, 86, 87
- ghypVar (Specific Generalized Hyperbolic Moments and Mode), 82
- gigCalcRange, 23
- gigChangePars, 8, 23, 24, 25, 34, 35, 84
- gigCheckPars, 26, 34, 35
- gigFit, 27, 32, 88, 89
- gigFitStart, 29, 30, 30
- gigFitStartLM (gigFitStart), 30
- gigFitStartMoM (gigFitStart), 30
- gigHessian, 32, 88
- gigKurt, 35
- gigKurt (Specific Generalized Inverse Gaussian Moments and Mode), 83
- gigLargeParam (gigParam), 36
- gigMean, 35
- gigMean (Specific Generalized Inverse Gaussian Moments and Mode), 83
- gigMode (Specific Generalized Inverse Gaussian Moments and Mode), 83
- gigMom, 33
- gigParam, 36
- GIGPlots, 37
- gigRawMom (gigMom), 33
- gigSkew, 35
- gigSkew (Specific Generalized Inverse Gaussian Moments and Mode), 83
- gigSmallParam (gigParam), 36
- gigVar, 35
- gigVar (Specific Generalized Inverse Gaussian Moments and Mode), 83
- hist, 29–31, 45–47, 51, 53, 70–72, 92
- hyperbCalcRange, 38
- hyperbChangePars, 5, 38, 39, 40, 56, 65–67, 86, 87
- hyperbCvMTest, 41, 59
- hyperbCvMTestPValue, 59
- hyperbCvMTestPValue (hyperbCvMTest), 41
- hyperbFit, 43, 47, 59, 90
- hyperbFitStand, 51–53

- hyperbFitStandStart, [52, 53](#)
- hyperbFitStart, [45, 46, 46, 80](#)
- hyperbFitStartMoM(hyperbFitStart), [46](#)
- hyperbHessian, [48, 90](#)
- hyperbKurt (Specific Hyperbolic Distribution Moments and Mode), [85](#)
- hyperbLargeParam(hyperbParam), [57](#)
- hyperbLargeShape(hyperbParam), [57](#)
- hyperblm, [49, 93](#)
- hyperblmFit, [51, 53](#)
- hyperbMean, [5](#)
- hyperbMean(Specific Hyperbolic Distribution Moments and Mode), [85](#)
- hyperbMode, [83](#)
- hyperbMode(Specific Hyperbolic Distribution Moments and Mode), [85](#)
- Hyperbolic, [54](#)
- hyperbParam, [57](#)
- HyperbPlots, [58](#)
- hyperbSkew(Specific Hyperbolic Distribution Moments and Mode), [85](#)
- hyperbSmallParam(hyperbParam), [57](#)
- hyperbSmallShape(hyperbParam), [57](#)
- hyperbVar(Specific Hyperbolic Distribution Moments and Mode), [85](#)
- hyperbWSqTable, [59](#)

- incompleteBesselK, [7](#)
- integrate, [8, 11, 12, 55, 56, 64, 65](#)
- is.wholenumber, [20, 34, 35](#)

- logHist, [9, 20, 30, 46, 70](#)

- mamquam, [60](#)
- MLambda(Functions for Moments), [4](#)
- model.matrix, [51](#)
- model.matrix.default, [50](#)
- momChangeAbout, [20, 34, 35](#)
- momIntegrated, [20, 35](#)
- momRecursion, [61](#)

- na.exclude, [50](#)
- na.fail, [50](#)
- na.omit, [50](#)

- nervePulse, [62](#)
- NIG, [63](#)
- nigCalcRange, [66](#)
- nigFit, [68, 72, 76, 94](#)
- nigFitStart, [69, 70, 71](#)
- nigFitStartMoM(nigFitStart), [71](#)
- nigHessian, [72, 94](#)
- nigKurt(Specific Normal Inverse Gaussian Distribution Moments and Mode), [86](#)
- nigLargeParam(nigParam), [74](#)
- nigLargeShape(nigParam), [74](#)
- nigMean(Specific Normal Inverse Gaussian Distribution Moments and Mode), [86](#)
- nigMode(Specific Normal Inverse Gaussian Distribution Moments and Mode), [86](#)
- nigParam, [74](#)
- nigPlots, [75](#)
- nigSkew(Specific Normal Inverse Gaussian Distribution Moments and Mode), [86](#)
- nigSmallParam(nigParam), [74](#)
- nigSmallShape(nigParam), [74](#)
- nigVar(Specific Normal Inverse Gaussian Distribution Moments and Mode), [86](#)
- nlm, [29, 45, 46, 53, 69, 70](#)
- nlminb, [45, 46](#)

- offset, [51](#)
- optim, [28–31, 44–47, 51, 53, 68–72, 83](#)
- options, [50](#)

- par, [29, 30, 44, 46, 69, 70](#)
- pghyp
(GeneralizedHyperbolicDistribution), [10](#)
- pgig(Generalized Inverse Gaussian), [5](#)
- phyperb(Hyperbolic), [54](#)
- plot.gigFit(gigFit), [27](#)
- plot.hyperbFit(hyperbFit), [43](#)
- plot.hyperblm, [53, 93](#)
- plot.hyperblm(hyperblm), [49](#)
- plot.nigFit(nigFit), [68](#)
- plotShapeTriangle, [76](#)
- pnig(NIG), [63](#)
- ppghyp(GeneralizedHyperbolicPlots), [13](#)

- ppgig, [30](#)
- ppgig (GIGPlots), [37](#)
- ppherb, [46](#)
- ppherb (HyperbPlots), [58](#)
- ppnig, [70](#)
- ppnig (nigPlots), [75](#)
- ppoints, [14](#), [38](#), [59](#), [76](#), [81](#)
- ppskewlap (SkewLaplacePlots), [80](#)
- print.gigFit, [89](#)
- print.gigFit (gigFit), [27](#)
- print.hyperbCvMTest (hyperbCvMTest), [41](#)
- print.hyperbFit, [90](#)
- print.hyperbFit (hyperbFit), [43](#)
- print.hyperblm, [53](#), [93](#)
- print.hyperblm (hyperblm), [49](#)
- print.nigFit, [94](#)
- print.nigFit (nigFit), [68](#)
- print.summary.gigFit (summary.gigFit), [88](#)
- print.summary.hyperbFit (summary.hyperbFit), [89](#)
- print.summary.hyperblm, [53](#), [93](#)
- print.summary.hyperblm (summary.hyperblm), [91](#)
- print.summary.nigFit (summary.nigFit), [93](#)
- pskewlap (SkewLaplace), [79](#)
- qghyp (GeneralizedHyperbolicDistribution), [10](#)
- qgig (Generalized Inverse Gaussian), [5](#)
- qhyperb (Hyperbolic), [54](#)
- qnig (NIG), [63](#)
- qqghyp (GeneralizedHyperbolicPlots), [13](#)
- qqgig, [30](#)
- qqgig (GIGPlots), [37](#)
- qqhyperb, [46](#)
- qqhyperb (HyperbPlots), [58](#)
- qqnig, [70](#)
- qqnig (nigPlots), [75](#)
- qqskewlap (SkewLaplacePlots), [80](#)
- qskewlap (SkewLaplace), [79](#)
- resistors, [77](#)
- rghyp (GeneralizedHyperbolicDistribution), [10](#)
- rgig (Generalized Inverse Gaussian), [5](#)
- rgig1 (Generalized Inverse Gaussian), [5](#)
- rhyperb (Hyperbolic), [54](#)
- RLambda, [83](#)
- RLambda (Functions for Moments), [4](#)
- rnig (NIG), [63](#)
- rskewlap (SkewLaplace), [79](#)
- safeIntegrate, [8](#), [12](#), [56](#), [64](#), [65](#)
- SandP500, [78](#)
- SkewLaplace, [79](#)
- SkewLaplacePlots, [80](#)
- SLambda (Functions for Moments), [4](#)
- Specific Generalized Hyperbolic Moments and Mode, [82](#)
- Specific Generalized Inverse Gaussian Moments and Mode, [83](#)
- Specific Hyperbolic Distribution Moments and Mode, [85](#)
- Specific Normal Inverse Gaussian Distribution Moments and Mode, [86](#)
- splinefun, [8](#), [12](#), [56](#), [65](#)
- summary, [89](#), [90](#), [94](#)
- summary.gigFit, [88](#)
- summary.hyperbFit, [89](#)
- summary.hyperblm, [53](#), [91](#)
- summary.nigFit, [93](#)
- sumX (hyperbHessian), [48](#)
- traffic, [95](#)
- tsHessian, [90](#), [91](#), [93](#)
- uniroot, [6](#), [8](#), [11](#), [12](#), [15](#), [24](#), [39](#), [55](#), [56](#), [64–66](#)
- vcov.gigFit (gigFit), [27](#)
- vcov.hyperbFit (hyperbFit), [43](#)
- vcov.nigFit (nigFit), [68](#)
- WLambda1 (Functions for Moments), [4](#)
- WLambda2 (Functions for Moments), [4](#)
- WLambda3 (Functions for Moments), [4](#)
- WLambda4 (Functions for Moments), [4](#)