Package ‘ghyp’

February 11, 2018

Type Package

Version 1.5.7

Date 2016-08-17

Title A Package on Generalized Hyperbolic Distribution and Its Special Cases

Author David Luethi, Wolfgang Breymann

Maintainer ORPHANED

LazyData no

Depends R(>= 2.7), methods, numDeriv, graphics, stats, gplots

Description Detailed functionality for working with the univariate and multivariate Generalized Hyperbolic distribution and its special cases (Hyperbolic (hyp), Normal Inverse Gaussian (NIG), Variance Gamma (VG), skewed Student-t and Gaussian distribution). Especially, it contains fitting procedures, an AIC-based model selection routine, and functions for the computation of density, quantile, probability, random variates, expected shortfall and some portfolio optimization and plotting routines as well as the likelihood ratio test. In addition, it contains the Generalized Inverse Gaussian distribution.

License GPL (>= 2)

Encoding latin1

Repository CRAN

Date/Publication 2016-08-17 11:01:16

NeedsCompilation yes

X-CRAN-Original-Maintainer Marc Weibel <marc.weibel@zhaw.ch>

X-CRAN-Comment Orphaned on 2018-02-11 as email address bounced. Same as FKF

R topics documented:

`ghyp-package` *A package on the generalized hyperbolic distribution and its special cases*

Description

This package provides detailed functionality for working with the univariate and multivariate Generalized Hyperbolic distribution and its special cases (Hyperbolic (hyp), Normal Inverse Gaussian (NIG), Variance Gamma (VG), skewed Student-t and Gaussian distribution). Especially, it contains fitting procedures, an AIC-based model selection routine, and functions for the computation of density, quantile, probability, random variates, expected shortfall and some portfolio optimization and plotting routines as well as the likelihood ratio test. In addition, it contains the Generalized Inverse Gaussian distribution.

Details

Package:	ghyp
Type:	Package
Version:	1.5.6
Date:	2013-02-04
License:	GPL (GNU Public Licence), Version 2 or later

Initialize:

ghyp	Initialize a generalized hyperbolic distribution.
hyp	Initialize a hyperbolic distribution.
NIG	Initialize a normal inverse gaussian distribution.
VG	Initialize a variance gamma distribution.
student.t	Initialize a Student-t distribution.
gauss	Initialize a Gaussian distribution.

Density, distribution function, quantile function and random generation:

dghyp	Density of a generalized hyperbolic distribution.
pghyp	Distribution function of a generalized hyperbolic distribution.
qghyp	Quantile of a univariate generalized hyperbolic distribution.
rghyp	Random generation of a generalized hyperbolic distribution.

Fit to data:

fit.ghypuv	Fit a generalized hyperbolic distribution to univariate data.
fit.hypuv	Fit a hyperbolic distribution to univariate data.
fit.NIGuv	Fit a normal inverse gaussian distribution to univariate data.
fit.VGuv	Fit a variance gamma distribution to univariate data.
fit.tuv	Fit a skewed Student-t distribution to univariate data.
fit.gaussuv	Fit a Gaussian distribution to univariate data.
fit.ghypmv	Fit a generalized hyperbolic distribution to multivariate data.
fit.hypmv	Fit a hyperbolic distribution to multivariate data.
fit.NIGmv	Fit a normal inverse gaussian distribution to multivariate data.
fit.VGmv	Fit a variance gamma distribution to multivariate data.
fit.tmv	Fit a skewed Student-t distribution to multivariate data.
fit.gaussmv	Fit a Gaussian distribution to multivariate data.
stepAIC.ghyp	Perform a model selection based on the AIC.

Risk, performance and portfolio optimization:

ESghyp	Expected shortfall of a univariate generalized hyperbolic distribution.

ghyp.omega	Performance measure *Omega* based on a univariate ghyp distribution.
portfolio.optimize	Calculate optimal portfolios with respect to alternative risk measures.

Utilities:

mean	Returns the expected value.
vcov	Returns the variance(-covariance).
ghyp.skewness	Skewness of a univariate ghyp distribution.
ghyp.kurtosis	Kurtosis of a univariate ghyp distribution.
logLik	Returns Log-Likelihood of fitted ghyp objects.
AIC	Returns the Akaike's Information Criterion of fitted ghyp objects.
lik.ratio.test	Performs a likelihood-ratio test on fitted ghyp distributions.
[	Extract certain dimensions of a multivariate ghyp distribution.
scale	Scale ghyp distribution objects to zero expectation and/or unit variance.
transform	Transform a multivariate generalized hyperbolic distribution.
ghyp.moment	Moments of the univariate ghyp distribution.
coef	Parameters of a generalized hyperbolic distribution.
ghyp.data	Data of a (fitted) generalized hyperbolic distribution.
ghyp.fit.info	Information about the fitting procedure, log-likelihood and AIC value.
ghyp.name	Returns the name of the ghyp distribution or a subclass of it.
ghyp.dim	Returns the dimension of a ghyp object.
summary	Summary of a fitted generalized hyperbolic distribution.

Plot functions:

qqghyp	Perform a quantile-quantile plot of a (fitted) univariate ghyp distribution.
hist	Plot a histogram of a (fitted) univariate generalized hyperbolic distribution.
pairs	Produce a matrix of scatterplots with quantile-quantile plots on the diagonal.
plot	Plot the density of a univariate ghyp distribution.
lines	Add the density of a univariate ghyp distribution to a graphics device.

Generalized inverse gaussian distribution:

dgig	Density of a generalized inverse gaussian distribution
pgig	Distribution function of a generalized inverse gaussian distribution
qgig	Quantile of a generalized inverse gaussian distribution
ESgig	Expected shortfall of a generalized inverse gaussian distribution
rgig	Random generation of a generalized inverse gaussian distribution

Package vignette:
A document about generalized hyperbolic distributions can be found in the doc folder of this package or on http://cran.r-project.org/package=ghyp.

Existing solutions

There are packages like GeneralizedHyperbolic, HyperbolicDist, SkewHyperbolic, VarianceGamma and fBasics which cover the univariate generalized hyperbolic distribution and/or some of its special cases. However, the univariate case is contained in this package as well because we aim to provide a uniform interface to deal with generalized hyperbolic distribution. Recently an R port of the S-Plus library QRMlib was released. The package QRMlib contains fitting procedures for the multivariate NIG, hyp and skewed Student-t distribution but not for the generalized hyperbolic case. The package fMultivar implements a fitting routine for multivariate skewed Student-t distributions as well.

Object orientation

We follow an object-oriented programming approach in this package and introduce distribution objects. There are mainly four reasons for that:

- Unlike most distributions the GH distribution has quite a few parameters which have to fulfill some consistency requirements. Consistency checks can be performed uniquely when an object is initialized.
- Once initialized the common functions belonging to a distribution can be called conveniently by passing the distribution object. A repeated input of the parameters is avoided.
- Distributions returned from fitting procedures can be directly passed to, e.g., the density function since fitted distribution objects add information to the distribution object and consequently inherit from the class of the distribution object.
- Generic method dispatching can be used to provide a uniform interface to, e.g., plot the probability density of a specific distribution like `plot(distribution.object)`. Additionally, one can take advantage of generic programming since R provides virtual classes and some forms of polymorphism.

Acknowledgement

This package has been partially developed in the framework of the COST-P10 “Physics of Risk” project. Financial support by the Swiss State Secretariat for Education and Research (SBF) is gratefully acknowledged.

Author(s)

David Luethi, Wolfgang Breymann

Institute of Data Analyses and Process Design (http://www.idp.zhaw.ch)

Maintainer: Marc Weibel <marc.weibel@zhaw.ch>

References

Quantitative Risk Management: Concepts, Techniques and Tools by Alexander J. McNeil, Ruediger Frey and Paul Embrechts
Princeton Press, 2005

Intermediate probability: A computational approach by Marc Paolella
Wiley, 2007

S-Plus and R *Library for Quantitative Risk Management QRMlib* by Alexander J. McNeil (2005) and Scott Ulman (R-port) (2007)
http://www.macs.hw.ac.uk/~mcneil/book/QRMlib.html and QRMlib

`coef-method` *Extract parameters of generalized hyperbolic distribution objects*

Description

The function `coef` returns the parameters of a generalized hyperbolic distribution object as a list. The user can choose between the "chi/psi", the "alpha.bar" and the "alpha/delta" parametrization. The function `coefficients` is a synonym for `coef`.

Usage

```
## S4 method for signature 'ghyp'
coef(object, type = c("chi.psi", "alpha.bar", "alpha.delta"))

## S4 method for signature 'ghyp'
coefficients(object, type = c("chi.psi", "alpha.bar", "alpha.delta"))
```

Arguments

`object`	An object inheriting from class ghyp.
`type`	According to `type` the parameters of either the "chi/psi", the "alpha.bar" or the "alpha/delta" parametrization will be returned. If `type` is missing, the parameters belonging to the parametrization of the construction are returned.

Details

Internally, the "chi/psi" parametrization is used. However, fitting is only possible in the "alpha.bar" parametrization as it provides the most convenient parameter constraints.

Value

If `type` is "chi.psi" a list with components:

`lambda`	Shape parameter.
`chi`	Shape parameter.
`psi`	Shape parameters.
`mu`	Location parameter.
`sigma`	Dispersion parameter.
`gamma`	Skewness parameter.

If `type` is “alpha.bar” a list with components:

`lambda`	Shape parameter.
`alpha.bar`	Shape parameter.
`mu`	Location parameter.
`sigma`	Dispersion parameter.
`gamma`	Skewness parameter.

If `type` is “alpha.delta” a list with components:

`lambda`	Shape parameter.
`alpha`	Shape parameter.
`delta`	Shape parameter.
`mu`	Location parameter.
`Delta`	Dispersion matrix with a determinant of 1 (only returned in the multivariate case).
`beta`	Shape and skewness parameter.

Note

A switch from either the “chi/psi” to the “alpha.bar” or from the “alpha/delta” to the “alpha.bar” parametrization is not yet possible.

Author(s)

David Luethi

See Also

`ghyp`, `fit.ghypuv`, `fit.ghypmv`, `ghyp.fit.info`, `transform`, `[.ghyp`

Examples

```
ghyp.mv <- ghyp(lambda = 1, alpha.bar = 0.1, mu = rep(0,2), sigma = diag(rep(1,2)),
                gamma = rep(0,2), data = matrix(rt(1000, df = 4), ncol = 2))
## Get parameters
coef(ghyp.mv, type = "alpha.bar")
coefficients(ghyp.mv, type = "chi.psi")

## Simple modification (do not modify slots directly e.g. object@mu <- 0:1)
param <- coef(ghyp.mv, type = "alpha.bar")
param$mu <- 0:1
do.call("ghyp", param) # returns a new 'ghyp' object
```

fit.ghypmv *Fitting generalized hyperbolic distributions to multivariate data*

Description

Perform a maximum likelihood estimation of the parameters of a multivariate generalized hyperbolic distribution by using an Expectation Maximization (EM) based algorithm.

Usage

```
fit.ghypmv(data, lambda = 1, alpha.bar = 1, mu = NULL, sigma = NULL,
           gamma = NULL, opt.pars = c(lambda = T, alpha.bar = T, mu = T,
                                      sigma = T, gamma = !symmetric),
           symmetric = F, standardize = F, nit = 2000, reltol = 1e-8,
           abstol = reltol * 10, na.rm = F, silent = FALSE, save.data = T,
           trace = TRUE, ...)

fit.hypmv(data,
          opt.pars = c(alpha.bar = T, mu = T, sigma = T, gamma = !symmetric),
          symmetric = F, ...)

fit.NIGmv(data,
          opt.pars = c(alpha.bar = T, mu = T, sigma = T, gamma = !symmetric),
          symmetric = F, ...)

fit.VGmv(data, lambda = 1,
         opt.pars = c(lambda = T, mu = T, sigma = T, gamma = !symmetric),
         symmetric = F, ...)

fit.tmv(data, nu = 3.5,
        opt.pars = c(lambda = T, mu = T, sigma = T, gamma = !symmetric),
        symmetric = F, ...)

fit.gaussmv(data, na.rm = T, save.data = T)
```

Arguments

data	An object coercible to a matrix.
lambda	Starting value for the shape parameter lambda.
alpha.bar	Starting value for the shape parameter alpha.bar.
nu	Starting value for the shape parameter nu (only used in case of a student-t distribution. It determines the degree of freedom and is defined as -2*lambda.)
mu	Starting value for the location parameter mu.
sigma	Starting value for the dispersion matrix sigma.
gamma	Starting value for the skewness vecotr gamma.

opt.pars	A named logical vector which states which parameters should be fitted.
symmetric	If TRUE the skewness parameter gamma keeps zero.
standardize	If TRUE the sample will be standardized before fitting. Afterwards, the parameters and log-likelihood et cetera will be back-transformed.
save.data	If TRUE data will be stored within the mle.ghyp object (cf. ghyp.data).
trace	If TRUE the evolution of the parameter values during the fitting procedure will be traced and stored (cf. ghyp.fit.info).
na.rm	If TRUE missing values will be removed from data.
silent	If TRUE no prompts will appear in the console.
nit	Maximal number of iterations of the expectation maximation algorithm.
reltol	Relative convergence tolerance.
abstol	Absolute convergence tolerance.
...	Arguments passed to optim and to fit.ghypmv when fitting special cases of the generalized hyperbolic distribution.

Details

This function uses a modified EM algorithm which is called Multi-Cycle Expectation Conditional Maximization (MCECM) algorithm. This algorithm is sketched in the vignette of this package which can be found in the doc folder. A more detailed description is provided by the book *Quantitative Risk Management, Concepts, Techniques and Tools* (see "References").

The general-purpose optimization routine optim is used to maximize the loglikelihood function of the univariate mixing distribution. The default method is that of Nelder and Mead which uses only function values. Parameters of optim can be passed via the ... argument of the fitting routines.

Value

An object of class mle.ghyp.

Note

The variance gamma distribution becomes singular when $\mathbf{x} - \mu = 0$. This singularity is catched and the reduced density function is computed. Because the transition is not smooth in the numerical implementation this can rarely result in nonsensical fits.

Providing both arguments, opt.pars and symmetric respectively, can result in a conflict when opt.pars['gamma'] and symmetric are TRUE. In this case symmetric will dominate and opt.pars['gamma'] is set to FALSE.

Author(s)

Wolfgang Breymann, David Luethi

References

Alexander J. McNeil, Ruediger Frey, Paul Embrechts (2005) *Quantitative Risk Management, Concepts, Techniques and Tools*

ghyp-package vignette in the doc folder or on http://cran.r-project.org/package=ghyp.

S-Plus and R library *QRMlib* (see http://www.macs.hw.ac.uk/~mcneil/book/QRMlib.html)

See Also

fit.ghypuv, fit.hypuv, fit.NIGuv, fit.VGuv, fit.tuv for univariate fitting routines. ghyp.fit.info for information regarding the fitting procedure.

Examples

```
data(smi.stocks)

fit.ghypmv(data = smi.stocks, opt.pars = c(lambda = FALSE), lambda = 2,
           control = list(rel.tol = 1e-5, abs.tol = 1e-5), reltol = 0.01)
```

---

fit.ghypuv *Fitting generalized hyperbolic distributions to univariate data*

Description

This function performs a maximum likelihood parameter estimation for univariate generalized hyperbolic distributions.

Usage

```
fit.ghypuv(data, lambda = 1, alpha.bar = 0.5, mu = median(data),
           sigma = mad(data), gamma = 0,
           opt.pars = c(lambda = T, alpha.bar = T, mu = T,
                        sigma = T, gamma = !symmetric),
           symmetric = F, standardize = F, save.data = T,
           na.rm = T, silent = FALSE, ...)

fit.hypuv(data,
          opt.pars = c(alpha.bar = T, mu = T, sigma = T, gamma = !symmetric),
          symmetric = F, ...)

fit.NIGuv(data,
          opt.pars = c(alpha.bar = T, mu = T, sigma = T, gamma = !symmetric),
          symmetric = F, ...)

fit.VGuv(data, lambda = 1,
```

```
          opt.pars = c(lambda = T, mu = T, sigma = T, gamma = !symmetric),
          symmetric = F, ...)

fit.tuv(data, nu = 3.5,
        opt.pars = c(nu = T, mu = T, sigma = T, gamma = !symmetric),
        symmetric = F, ...)

fit.gaussuv(data, na.rm = T, save.data = T)
```

Arguments

`data`	An object coercible to a `vector`.
`lambda`	Starting value for the shape parameter `lambda`.
`alpha.bar`	Starting value for the shape parameter `alpha.bar`.
`nu`	Starting value for the shape parameter `nu` (only used in case of a student-t distribution. It determines the degree of freedom and is defined as `-2*lambda`.)
`mu`	Starting value for the location parameter `mu`.
`sigma`	Starting value for the dispersion parameter `sigma`.
`gamma`	Starting value for the skewness parameter `gamma`.
`opt.pars`	A named logical `vector` which states which parameters should be fitted.
`symmetric`	If `TRUE` the skewness parameter `gamma` keeps zero.
`standardize`	If `TRUE` the sample will be standardized before fitting. Afterwards, the parameters and log-likelihood et cetera will be back-transformed.
`save.data`	If `TRUE` `data` will be stored within the `mle.ghyp` object.
`na.rm`	If `TRUE` missing values will be removed from `data`.
`silent`	If `TRUE` no prompts will appear in the console.
`...`	Arguments passed to `optim` and to `fit.ghypuv` when fitting special cases of the generalized hyperbolic distribution.

Details

The general-purpose optimization routine `optim` is used to maximize the loglikelihood function. The default method is that of Nelder and Mead which uses only function values. Parameters of `optim` can be passed via the ... argument of the fitting routines.

Value

An object of class `mle.ghyp`.

Note

The variance gamma distribution becomes singular when $x - \mu = 0$. This singularity is catched and the reduced density function is computed. Because the transition is not smooth in the numerical implementation this can rarely result in nonsensical fits.

Providing both arguments, `opt.pars` and `symmetric` respectively, can result in a conflict when `opt.pars['gamma']` and `symmetric` are `TRUE`. In this case `symmetric` will dominate and `opt.pars['gamma']` is set to `FALSE`.

Author(s)

Wolfgang Breymann, David Luethi

References

ghyp-package vignette in the doc folder or on http://cran.r-project.org/package=ghyp.

See Also

fit.ghypmv, fit.hypmv, fit.NIGmv, fit.VGmv, fit.tmv for multivariate fitting routines. ghyp.fit.info for information regarding the fitting procedure.

Examples

```
data(smi.stocks)

nig.fit <- fit.NIGuv(smi.stocks[,"SMI"], opt.pars = c(alpha.bar = FALSE),
                     alpha.bar = 1, control = list(abstol = 1e-8))
nig.fit

summary(nig.fit)

hist(nig.fit)
```

---

ghyp-constructors *Create generalized hyperbolic distribution objects*

Description

Constructor functions for univariate and multivariate generalized hyperbolic distribution objects and their special cases in one of the parametrizations “chi/psi”, “alpha.bar” and “alpha/delta”.

Usage

```
ghyp(lambda = 0.5, chi = 0.5, psi = 2, mu = 0, sigma = diag(rep(1, length(mu))),
     gamma = rep(0, length(mu)), alpha.bar = NULL, data = NULL)

ghyp.ad(lambda = 0.5, alpha = 1.5, delta = 1, beta = rep(0, length(mu)),
        mu = 0, Delta = diag(rep(1, length(mu))), data = NULL)


hyp(chi = 0.5, psi = 2, mu = 0, sigma = diag(rep(1, length(mu))),
    gamma = rep(0, length(mu)), alpha.bar = NULL, data = NULL)

hyp.ad(alpha = 1.5, delta = 1, beta = rep(0, length(mu)), mu = 0,
       Delta = diag(rep(1, length(mu))), data = NULL)
```

```
NIG(chi = 2, psi = 2, mu = 0, sigma = diag(rep(1, length(mu))),
    gamma = rep(0, length(mu)), alpha.bar = NULL, data = NULL)

NIG.ad(alpha = 1.5, delta = 1, beta = rep(0, length(mu)), mu = 0,
       Delta = diag(rep(1, length(mu))), data = NULL)


student.t(nu = 3.5, chi = nu - 2, mu = 0, sigma = diag(rep(1, length(mu))),
          gamma = rep(0, length(mu)), data = NULL)

student.t.ad(lambda = -2, delta = 1, beta = rep(0, length(mu)), mu = 0,
             Delta = diag(rep(1, length(mu))), data = NULL)


VG(lambda = 1, psi = 2*lambda, mu = 0, sigma = diag(rep(1, length(mu))),
   gamma = rep(0, length(mu)), data = NULL)

VG.ad(lambda = 2, alpha = 1.5, beta = rep(0, length(mu)), mu = 0,
      Delta = diag(rep(1, length(mu))), data = NULL)

gauss(mu = 0, sigma = diag(rep(1, length(mu))), data = NULL)
```

Arguments

`lambda`	Shape parameter. Common for all parametrizations.
`nu`	Shape parameter only used in case of a Student-t distribution in the “chi/psi” and “alpha.bar” parametrization . It determines the degree of freedom.
`chi`	Shape parameter of the “chi/psi” parametrization.
`psi`	Shape parameter of the “chi/psi” parametrization.
`alpha`	Shape parameter of the “alpha/delta” parametrization.
`delta`	Shape parameter of the “alpha/delta” parametrization.
`alpha.bar`	Shape parameter of the “alpha.bar” parametrization. Supplying “alpha.bar” makes the parameters “chi” and “psi” redundant.
`mu`	Location parameter. Either a scalar or a vector. Common for all parametrizations.
`sigma`	Dispersion parameter of the “chi/psi” parametrization. Either a scalar or a matrix.
`Delta`	Dispersion parameter. Must be a matrix with a determinant of 1. This parameter is only used in the multivariate case of the “alpha.beta” parametrization.
`gamma`	Skewness parameter of the “chi/psi” parametrization. Either a scalar or a vector.
`beta`	Skewness parameter of the “alpha/delta” parametrization. Either a scalar or a vector.
`data`	An object coercible to a `vector` (univariate case) or `matrix` (multivariate case).

Details

These functions serve as constructors for univariate and multivariate objects.

ghyp, hyp and NIG are constructor functions for both the “chi/psi” and the “alpha.bar” parametrization. Whenever alpha.bar is not NULL it is assumed that the “alpha.bar” parametrization is used and the parameters “chi” and “psi” become redundant.

Similarly, the variance gamma (VG) and the Student-t distribution share the same constructor function for both the chi/psi and alpha.bar parametrization. To initialize them in the alpha.bar parametrization simply omit the argument psi and chi, respectively. If psi or chi are submitted, the “chi/psi” parametrization will be used.

ghyp.ad, hyp.ad, NIG.ad, student.t.ad and VG.ad use the “alpha/delta” parametrization.

The following table gives the constructors for each combination of distribution and parametrization.

		Parametrization	
Distribution	“chi/psi”	“alpha.bar”	“alpha/delta”
GH	ghyp(...)	ghyp(..., alpha.bar=x)	ghyp.ad(...)
hyp	hyp(...)	hyp(..., alpha.bar=x)	hyp.ad(...)
NIG	NIG(...)	NIG(..., alpha.bar=x)	NIG.ad(...)
Student-t	student.t(..., chi=x)	student.t(...)	student.t.ad(...)
VG	VG(..., psi=x)	VG(...)	VG.ad(...)

Have a look on the vignette of this package in the doc folder for further information regarding the parametrization and for the domains of variation of the parameters.

Value

An object of class ghyp.

Note

The Student-t parametrization obtained via the “alpha.bar” parametrization slightly differs from the common Student-t parametrization: The parameter sigma denotes the standard deviation in the univariate case and the variance in the multivariate case. Thus, set $\sigma = \sqrt{\nu/(\nu-2)}$ in the univariate case to get the same results as with the standard R implementation of the Student-t distribution.

In case of non-finite variance, the “alpha.bar” parametrization does not work because sigma is defined to be the standard deviation. In this case the “chi/psi” parametrization can be used by submitting the parameter chi. To obtain equal results as the standard R implmentation use student.t(nu = nu, chi = nu) (see **Examples**).

Have a look on the vignette of this package in the doc folder for further information.

Once an object of class ghyp is created the methods Xghyp have to be used even when the distribution is a special case of the GH distribution. E.g. do not use dVG. Use dghyp and submit a variance gamma distribution created with VG().

Author(s)

David Luethi

References

ghyp-package vignette in the doc folder or on http://cran.r-project.org/package=ghyp

See Also

ghyp-class for a summary of generic methods assigned to ghyp objects, coef for switching between different parametrizations, d/p/q/r/ES/gyhp for density, distribution function et cetera, fit.ghypuv and fit.ghypmv for fitting routines.

Examples

```
## alpha.bar parametrization of a univariate GH distribution
ghyp(lambda=2, alpha.bar=0.1, mu=0, sigma=1, gamma=0)
## lambda/chi parametrization of a univariate GH distribution
ghyp(lambda=2, chi=1, psi=0.5, mu=0, sigma=1, gamma=0)
## alpha/delta parametrization of a univariate GH distribution
ghyp.ad(lambda=2, alpha=0.5, delta=1, mu=0, beta=0)

## alpha.bar parametrization of a multivariate GH distribution
ghyp(lambda=1, alpha.bar=0.1, mu=2:3, sigma=diag(1:2), gamma=0:1)
## lambda/chi parametrization of a multivariate GH distribution
ghyp(lambda=1, chi=1, psi=0.5, mu=2:3, sigma=diag(1:2), gamma=0:1)
## alpha/delta parametrization of a multivariate GH distribution
ghyp.ad(lambda=1, alpha=2.5, delta=1, mu=2:3, Delta=diag(c(1,1)), beta=0:1)

## alpha.bar parametrization of a univariate hyperbolic distribution
hyp(alpha.bar=0.3, mu=1, sigma=0.1, gamma=0)
## lambda/chi parametrization of a univariate hyperbolic distribution
hyp(chi=1, psi=2, mu=1, sigma=0.1, gamma=0)
## alpha/delta parametrization of a univariate hyperbolic distribution
hyp.ad(alpha=0.5, delta=1, mu=0, beta=0)

## alpha.bar parametrization of a univariate NIG distribution
NIG(alpha.bar=0.3, mu=1, sigma=0.1, gamma=0)
## lambda/chi parametrization of a univariate NIG distribution
NIG(chi=1, psi=2, mu=1, sigma=0.1, gamma=0)
## alpha/delta parametrization of a univariate NIG distribution
NIG.ad(alpha=0.5, delta=1, mu=0, beta=0)

## alpha.bar parametrization of a univariate VG distribution
VG(lambda=2, mu=1, sigma=0.1, gamma=0)
## alpha/delta parametrization of a univariate VG distribution
VG.ad(lambda=2, alpha=0.5, mu=0, beta=0)
```

```
## alpha.bar parametrization of a univariate t distribution
student.t(nu = 3, mu=1, sigma=0.1, gamma=0)
## alpha/delta parametrization of a univariate t distribution
student.t.ad(lambda=-2, delta=1, mu=0, beta=1)

## Obtain equal results as with the R-core parametrization
## of the t distribution:
nu <- 4
standard.R.chi.psi <- student.t(nu = nu, chi = nu)
standard.R.alpha.bar <- student.t(nu = nu, sigma = sqrt(nu  /(nu - 2)))

random.sample <- rnorm(3)
dt(random.sample, nu)
dghyp(random.sample, standard.R.chi.psi)   # all implementations yield...
dghyp(random.sample, standard.R.alpha.bar) # ...the same values

random.quantiles <- runif(4)
qt(random.quantiles, nu)
qghyp(random.quantiles, standard.R.chi.psi)   # all implementations yield...
qghyp(random.quantiles, standard.R.alpha.bar) # ...the same values

## If nu <= 2 the "alpha.bar" parametrization does not exist, but the
## "chi/psi" parametrization. The case of a Cauchy distribution:
nu <- 1
standard.R.chi.psi <- student.t(nu = nu, chi = nu)

dt(random.sample, nu)
dghyp(random.sample, standard.R.chi.psi)   # both give the same result

pt(random.sample, nu)
pghyp(random.sample, standard.R.chi.psi) # both give the same result
```

`ghyp-distribution` *The Generalized Hyperbolic Distribution*

Description

Density, distribution function, quantile function, expected-shortfall and random generation for the univariate and multivariate generalized hyperbolic distribution and its special cases.

Usage

```
dghyp(x, object = ghyp(), logvalue = FALSE)

pghyp(q, object = ghyp(), n.sim = 10000, subdivisions = 200,
      rel.tol = .Machine$double.eps^0.5, abs.tol = rel.tol,
      lower.tail = TRUE)
```

```
qghyp(p, object = ghyp(), method = c("integration", "splines"),
      spline.points = 200, subdivisions = 200,
      root.tol = .Machine$double.eps^0.5,
      rel.tol = root.tol^1.5, abs.tol = rel.tol)

rghyp(n, object = ghyp())
```

Arguments

`p`	A vector of probabilities.
`x`	A vector, matrix or data.frame of quantiles.
`q`	A vector, matrix or data.frame of quantiles.
`n`	Number of observations.
`object`	An object inheriting from class `ghyp`.
`logvalue`	If `TRUE` the logarithm of the density will be returned.
`n.sim`	The number of simulations when computing pghyp of a multivariate generalized hyperbolic distribution.
`subdivisions`	The number of subdivisions passed to `integrate` when computing the distribution function pghyp of a univariate generalized hyperbolic distribution.
`rel.tol`	The relative accuracy requested from `integrate`.
`abs.tol`	The absolute accuracy requested from `integrate`.
`lower.tail`	If TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`method`	The method how quantiles are computed (see **Details**).
`spline.points`	The number of support points when computing the quantiles with the method “splines” instead of “integration”.
`root.tol`	The tolerance of `uniroot`.

Details

qghyp only works for univariate generalized hyperbolic distributions.

pghyp performs a numeric integration of the density in the univariate case. The multivariate cumulative distribution is computed by means of monte carlo simulation.

qghyp computes the quantiles either by using the “integration” method where the root of the distribution function is solved or via “splines” which interpolates the distribution function and solves it with `uniroot` afterwards. The “integration” method is recommended when only few quantiles are required. If more than approximately 20 quantiles are needed to be calculated the “splines” method becomes faster. The accuracy can be controlled with an adequate setting of the parameters `rel.tol`, `abs.tol`, `root.tol` and `spline.points`.

rghyp uses the random generator for generalized inverse Gaussian distributed random variates from the Rmetrics package *fBasics* (cf. `rgig`).

Value

dghyp gives the density,
pghyp gives the distribution function,
qghyp gives the quantile function,
rghyp generates random deviates.

Note

Objects generated with hyp, NIG, VG and student.t have to use Xghyp as well. E.g. dNIG(0, NIG()) does not work but dghyp(0, NIG()).

When the skewness becomes very large the functions using qghyp may fail. The functions qqghyp, pairs and portfolio.optimize are based on qghyp.

Author(s)

David Luethi

References

ghyp-package vignette in the doc folder or on http://cran.r-project.org/package=ghyp and references therein.

See Also

ghyp-class definition, ghyp constructors, fitting routines fit.ghypuv and fit.ghypmv, risk and performance measurement ESghyp and ghyp.omega, transformation and subsettting of ghyp objects, integrate, spline.

Examples

```
## Univariate generalized hyperbolic distribution
univariate.ghyp <- ghyp()

par(mfrow=c(5, 1))

quantiles <- seq(-4, 4, length = 500)
plot(quantiles, dghyp(quantiles, univariate.ghyp))
plot(quantiles, pghyp(quantiles, univariate.ghyp))

probabilities <- seq(1e-4, 1-1e-4, length = 500)
plot(probabilities, qghyp(probabilities, univariate.ghyp, method = "splines"))

hist(rghyp(n=10000,univariate.ghyp),nclass=100)

## Mutivariate generalized hyperbolic distribution
multivariate.ghyp <- ghyp(sigma=var(matrix(rnorm(10),ncol=2)),mu=1:2,gamma=-(2:1))

par(mfrow=c(2, 1))
```

```
quantiles <- outer(seq(-4, 4, length = 50), c(1, 1))
plot(quantiles[, 1], dghyp(quantiles, multivariate.ghyp))
plot(quantiles[, 1], pghyp(quantiles, multivariate.ghyp, n.sim = 1000))

rghyp(n = 10, multivariate.ghyp)
```

---

ghyp-get *Get methods for objects inheriting from class ghyp*

Description

These functions simply return data stored within generalized hyperbolic distribution objects, i.e. slots of the classes ghyp and mle.ghyp. ghyp.fit.info extracts information about the fitting procedure from objects of class mle.ghyp. ghyp.data returns the data slot of a gyhp object. ghyp.dim returns the dimension of a gyhp object. ghyp.name returns the name of the distribution of a gyhp object.

Usage

```
ghyp.fit.info(object)

ghyp.data(object)

ghyp.name(object, abbr = FALSE, skew.attr = TRUE)

ghyp.dim(object)
```

Arguments

object	An object inheriting from class ghyp.
abbr	If TRUE the abbreviation of the ghyp distribution will be returned.
skew.attr	If TRUE an attribute will be added to the name of the ghyp distribution stating whether the distribution is symmetric or not.

Value

ghyp.fit.info returns list with components:

logLikelihood	The maximized log-likelihood value.
aic	The Akaike information criterion.
fitted.params	A boolean vector stating which parameters were fitted.
converged	A boolean whether optim converged or not.
n.iter	The number of iterations.
error.code	Error code from optim.
error.message	Error message from optim.
parameter.variance	Parameter variance (only for univariate fits).
trace.pars	Trace values of the parameters during the fitting procedure.

ghyp.data returns NULL if no data is stored within the object, a vector if it is an univariate generalized hyperbolic distribution and matrix if it is an multivariate generalized hyperbolic distribution.

ghyp.name returns the name of the ghyp distribution which can be the name of a special case. Depending on the arguments abbr and skew.attr one of the following is returned.

abbr == FALSE & skew.attr == TRUE	abbr == TRUE & skew.attr == TRUE
(A)symmetric Generalized Hyperbolic	(A)symm ghyp
(A)symmetric Hyperbolic	(A)symm hyp
(A)symmetric Normal Inverse Gaussian	(A)symm NIG
(A)symmetric Variance Gamma	(A)symm VG
(A)symmetric Student-t	(A)symm t
Gaussian	Gauss
abbr == FALSE & skew.attr == FALSE	abbr == TRUE & skew.attr == FALSE
Generalized Hyperbolic	ghyp
Hyperbolic	hyp
Normal Inverse Gaussian	NIG
Variance Gamma	VG
Student-t	t
Gaussian	Gauss

ghyp.dim returns the dimension of a ghyp object.

Note

ghyp.fit.info requires an object of class mle.ghyp. In the univariate case the parameter variance is returned as well. The parameter variance is defined as the inverse of the negative hesse-matrix computed by optim. Note that this makes sense only in the case that the estimates are asymptotically normal distributed.

The class ghyp contains a data slot. Data can be stored either when an object is initialized or via the fitting routines and the argument save.data.

Author(s)

David Luethi

See Also

coef, mean, vcov, logLik, AIC for other accessor functions, fit.ghypmv, fit.ghypuv, ghyp for constructor functions, optim for possible error messages.

Examples

```
## multivariate generalized hyperbolic distribution
ghyp.mv <- ghyp(lambda = 1, alpha.bar = 0.1, mu = rep(0, 2), sigma = diag(rep(1, 2)),
                gamma = rep(0, 2), data = matrix(rt(1000, df = 4), ncol = 2))
```

```
## Get data
ghyp.data(ghyp.mv)

## Get the dimension
ghyp.dim(ghyp.mv)

## Get the name of the ghyp object
ghyp.name(ghyp(alpha.bar = 0))
ghyp.name(ghyp(alpha.bar = 0, lambda = -4), abbr = TRUE)

## 'ghyp.fit.info' does only work when the object is of class 'mle.ghyp',
## i.e. is created by 'fit.ghypuv' etc.
mv.fit <- fit.tmv(data = ghyp.data(ghyp.mv), control = list(abs.tol = 1e-3))
ghyp.fit.info(mv.fit)
```

---

ghyp-mle.ghyp-classes *Classes ghyp and mle.ghyp*

Description

The class "ghyp" basically contains the parameters of a generalized hyperbolic distribution. The class "mle.ghyp" inherits from the class "ghyp". The class "mle.ghyp" adds some additional slots which contain information about the fitting procedure. Namely, these are the number of iterations (`n.iter`), the log likelihood value (`llh`), the Akaike Information Criterion (`aic`), a boolean vector (`fitted.params`) stating which parameters were fitted, a boolean `converged` whether the fitting procedure converged or not, an `error.code` which stores the status of a possible error and the corresponding `error.message`. In the univariate case the parameter variance is also stored in `parameter.variance`.

Objects from the Class

Objects should only be created by calls to the constructors `ghyp`, `hyp`, `NIG`, `VG`, `student.t` and `gauss` or by calls to the fitting routines like `fit.ghypuv`, `fit.ghypmv`, `fit.hypuv`, `fit.hypmv` et cetera.

Slots

Slots of class ghyp:

`call`: The function-call of class `call`.

`lambda`: Shape parameter of class `numeric`.

`alpha.bar`: Shape parameter of class `numeric`.

`chi`: Shape parameter of an alternative parametrization. Object of class `numeric`.

`psi`: Shape parameter of an alternative parametrization. Object of class `numeric`.

`mu`: Location parameter of lass `numeric`.

`sigma`: Dispersion parameter of class `matrix`.

`gamma`: Skewness parameter of class `numeric`.

`model`: Model, i.e., (a)symmetric generalized hyperbolic distribution or (a)symmetric special case. Object of class `character`.

`dimension`: Dimension of the generalized hyperbolic distribution. Object of class `numeric`.

`expected.value`: The expected value of a generalized hyperbolic distribution. Object of class `numeric`.

`variance`: The variance of a generalized hyperbolic distribution of class `matrix`.

`data`: The data-slot is of class `matrix`. When an object of class `ghypmv` is instantiated the user can decide whether data should be stored within the object or not. This is the default and may be useful when fitting eneralized hyperbolic distributions to data and perform further analysis afterwards.

`parametrization`: Parametrization of the generalized hyperbolic distribution of class `character`. These are currently either "chi.psi", "alpha.bar" or "alpha.delta".

Slots added by class mle.ghyp:

`n.iter`: The number of iterations of class `numeric`.

`llh`: The log likelihood value of class `numeric`.

`converged`: A boolean whether converged or not. Object of class `logical`.

`error.code`: An error code of class `numeric`.

`error.message`: An error message of class `character`.

`fitted.params`: A boolean vector stating which parameters were fitted of class `logical`.

`aic`: The value of the Akaike Information Criterion of class `numeric`.

`parameter.variance`: The parameter variance is the inverse of the fisher information matrix. This slot is filled only in the case of an univariate fit. This slot is of class `matrix`.

`trace.pars`: Contains the parameter value evolution during the fitting procedure. `trace.pars` of class `list`.

Extends

Class "mle.ghyp" extends class `"ghyp"`, directly.

Methods

A "pairs" method (see `pairs`).
A "hist" method (see `hist`).
A "plot" method (see `plot`).
A "lines" method (see `lines`).
A "coef" method (see `coef`).
A "mean" method (see `mean`).
A "vcov" method (see `vcov`).
A "scale" method (see `scale`).
A "transform" method (see `transform`).
A "[.ghyp" method (see `[`).
A "logLik" method for objects of class "mle.ghyp" (see `logLik`).
An "AIC" method for objects of class "mle.ghyp" (see `AIC`).
A "summary" method for objects of class "mle.ghyp" (see `summary`).

Note

When showing special cases of the generalized hyperbolic distribution the corresponding fixed parameters are not printed.

Author(s)

David Luethi

See Also

`optim` for an interpretation of `error.code`, `error.message` and `parameter.variance`.
`ghyp`, `hyp`, `NIG`, `VG`, `student.t` and `gauss` for constructors of the class ghyp in the "alpha.bar" and "chi/psi" parametrization. `xxx.ad` for all the constructors in the "alpha/delta" parametrization. `fit.ghypuv`, `fit.ghypmv` et cetera for the fitting routies and constructors of the class `mle.ghyp`.

Examples

```
data(smi.stocks)
multivariate.fit <- fit.ghypmv(data = smi.stocks,
                               opt.pars = c(lambda = FALSE, alpha.bar = FALSE),
                               lambda = 2)
summary(multivariate.fit)

vcov(multivariate.fit)
mean(multivariate.fit)
logLik(multivariate.fit)
AIC(multivariate.fit)
coef(multivariate.fit)

univariate.fit <- multivariate.fit[1]
hist(univariate.fit)

plot(univariate.fit)
lines(multivariate.fit[2])
```

---

ghyp-risk-performance *Risk and Performance Measures*

Description

Functions to compute the risk measure *Expected Shortfall* and the performance measure *Omega* based on univariate generalized hyperbolic distributions.

Usage

```
ESghyp(alpha, object = ghyp(), distr = c("return", "loss"), ...)

ghyp.omega(L, object = ghyp(), ...)
```

Arguments

`alpha`	A vector of confidence levels.
`L`	A vector of threshold levels.
`object`	A univarite generalized hyperbolic distribution object inheriting from class `ghyp`.
`distr`	Whether the ghyp-object specifies a return or a loss-distribution (see **Details**).
`...`	Arguments passed from `ESghyp` to `qghyp` and from `ghyp.omega` `integrate`.

Details

The parameter `distr` specifies whether the ghyp-object describes a return or a loss-distribution. In case of a return distribution the expected-shortfall on a confidence level α is defined as $\mathrm{ES}_\alpha := \mathrm{E}(X|X \leq F_X^{-1}(\alpha))$ while in case of a loss distribution it is defined on a confidence level α as $\mathrm{ES}_\alpha := \mathrm{E}(X|X > F_X^{-1}(\alpha))$.

Omega is defined as the ratio of a European call-option price divided by a put-option price with strike price L (see **References**): $\Omega(L) := \frac{C(L)}{P(L)}$.

Value

`ESghyp` gives the expected shortfall and
`ghyp.omega` gives the performance measure Omega.

Author(s)

David Luethi

References

Omega as a Performance Measure by Hossein Kazemi, Thomas Schneeweis and Raj Gupta
University of Massachusetts, 2003

See Also

`ghyp-class` definition, `ghyp` constructors, univariate fitting routines, `fit.ghypuv`, `portfolio.optimize` for portfolio optimization with respect to alternative risk measures, `integrate`.

Examples

```
data(smi.stocks)

## Fit a NIG model to Credit Suisse and Swiss Re log-returns
cs.fit <- fit.NIGuv(smi.stocks[, "CS"], silent = TRUE)
swiss.re.fit <- fit.NIGuv(smi.stocks[, "Swiss.Re"], silent = TRUE)

## Confidence levels for expected shortfalls
es.levels <- c(0.001, 0.01, 0.05, 0.1)
```

```
cs.es <- ESghyp(es.levels, cs.fit)
swiss.re.es <- ESghyp(es.levels, swiss.re.fit)

## Threshold levels for Omega
threshold.levels <- c(0, 0.01, 0.02, 0.05)

cs.omega <- ghyp.omega(threshold.levels, cs.fit)
swiss.re.omega <- ghyp.omega(threshold.levels, swiss.re.fit)

par(mfrow = c(2, 1))

barplot(rbind(CS = cs.es, Swiss.Re = swiss.re.es), beside = TRUE,
        names.arg = paste(100 * es.levels, "percent"), col = c("gray40", "gray80"),
        ylab = "Expected Shortfalls (return distribution)", xlab = "Level")

legend("bottomright", legend = c("CS", "Swiss.Re"), fill = c("gray40", "gray80"))

barplot(rbind(CS = cs.omega, Swiss.Re = swiss.re.omega), beside = TRUE,
        names.arg = threshold.levels, col = c("gray40", "gray80"),
        ylab = "Omega", xlab = "Threshold level")
legend("topright", legend = c("CS", "Swiss.Re"), fill = c("gray40", "gray80"))

## => the higher the performance, the higher the risk (as it should be)
```

`ghyp.moment` *Compute moments of generalized hyperbolic distributions*

Description

This function computes moments of arbitrary orders of the univariate generalized hyperbolic distribution. The expectation of $f(X-c)^k$ is calculated. f can be either the absolute value or the identity. c can be either zero or $E(X)$.

Usage

```
ghyp.moment(object, order = 3:4, absolute = FALSE, central = TRUE, ...)
```

Arguments

`object`	A univarite generalized hyperbolic object inheriting from class `ghyp`.
`order`	A vector containing the order of the moments.
`absolute`	Indicate whether the absolute value is taken or not. If `absolute = TRUE` then $E(\|X-c\|^k)$ is computed. Otherwise $E((X-c)^k)$. c depends on the argument `central`. `absolute` must be `TRUE` if `order` is not integer.
`central`	If `TRUE` the moment around the expected value $E((X-E(X))^k)$ is computed. Otherwise $E(X^k)$.
`...`	Arguments passed to `integrate`.

Details

In general `ghyp.moment` is based on numerical integration. For the special cases of either a "ghyp", "hyp" or "NIG" distribution analytic expressions (see **References**) will be taken if non-absolute and non-centered moments of integer order are requested.

Value

A vector containing the moments.

Author(s)

David Luethi

References

Moments of the Generalized Hyperbolic Distribution by David J. Scott, Diethelm Wuertz and Thanh Tam Tran
Working paper, 2008

See Also

`mean`, `vcov`, `Egig`

Examples

```
nig.uv <- NIG(alpha.bar = 0.1, mu = 1.1, sigma = 3, gamma = -2)

# Moments of integer order
ghyp.moment(nig.uv, order = 1:6)

# Moments of fractional order
ghyp.moment(nig.uv, order = 0.2 * 1:20, absolute = TRUE)
```

`gig-distribution` *The Generalized Inverse Gaussian Distribution*

Description

Density, distribution function, quantile function, random generation, expected shortfall and expected value and variance for the generalized inverse gaussian distribution.

Usage

```
dgig(x, lambda = 1, chi = 1, psi = 1, logvalue = FALSE)

pgig(q, lambda = 1, chi = 1, psi = 1, ...)

qgig(p, lambda = 1, chi = 1, psi = 1, method = c("integration", "splines"),
     spline.points = 200, subdivisions = 200,
     root.tol = .Machine$double.eps^0.5,
     rel.tol = root.tol^1.5, abs.tol = rel.tol, ...)

rgig(n = 10, lambda = 1, chi = 1, psi = 1)

ESgig(alpha, lambda = 1, chi = 1, psi = 1, distr = c("return", "loss"), ...)

Egig(lambda, chi, psi, func = c("x", "logx", "1/x", "var"), check.pars = TRUE)
```

Arguments

`x`	A vector of quantiles.
`q`	A vector of quantiles.
`p`	A vector of probabilities.
`alpha`	A vector of confidence levels.
`n`	Number of observations.
`lambda`	A shape and scale and parameter.
`chi, psi`	Shape and scale parameters. Must be positive.
`logvalue`	If `TRUE` the logarithm of the density will be returned.
`distr`	Whether the ghyp-object specifies a return or a loss-distribution (see **Details**).
`subdivisions`	The number of subdivisions passed to `integrate` when computing the the distribution function `pgig`.
`rel.tol`	The relative accuracy requested from `integrate`.
`abs.tol`	The absolute accuracy requested from `integrate`.
`method`	Determines which method is used when calculating quantiles.
`spline.points`	The number of support points when computing the quantiles with the method "splines" instead of "integration".
`root.tol`	The tolerance of `uniroot`.
`func`	The transformation function when computing the expected value. `x` is the expected value (default), `log x` returns the expected value of the logarithm of `x`, `1/x` returns the expected value of the inverse of `x` and `var` returns the variance.
`check.pars`	If `TRUE` the parameters are checked first.
`...`	Arguments passed form `ESgig` to `qgig`.

Details

qgig computes the quantiles either by using the “integration” method where the root of the distribution function is solved or via “splines” which interpolates the distribution function and solves it with uniroot afterwards. The “integration” method is recommended when few quantiles are required. If more than approximately 20 quantiles are needed to be calculated the “splines” method becomes faster. The accuracy can be controlled with an adequate setting of the parameters rel.tol, abs.tol, root.tol and spline.points.

rgig relies on the C function with the same name kindly provided by Ester Pantaleo and Robert B. Gramacy.

Egig with func = "log x" uses grad from the R package *numDeriv*. See the package vignette for details regarding the expectation of GIG random variables.

Value

dgig gives the density,
pgig gives the distribution function,
qgig gives the quantile function,
ESgig gives the expected shortfall,
rgig generates random deviates and
Egig gives the expected value of either x, 1/x, log(x) or the variance if func equals var.

Author(s)

David Luethi and Ester Pantaleo

References

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

Michael, J. R, Schucany, W. R, Haas, R, W. (1976). *Generating random variates using transformations with multiple roots*, The American Statistican, **30**, 88–90.

See Also

fit.ghypuv, fit.ghypmv, integrate, uniroot, spline

Examples

```
dgig(1:40, lambda = 10, chi = 1, psi = 1)
qgig(1e-5, lambda = 10, chi = 1, psi = 1)

ESgig(c(0.19,0.3), lambda = 10, chi = 1, psi = 1, distr = "loss")
ESgig(alpha=c(0.19,0.3), lambda = 10, chi = 1, psi = 1, distr = "ret")

Egig(lambda = 10, chi = 1, psi = 1, func = "x")
Egig(lambda = 10, chi = 1, psi = 1, func = "var")
Egig(lambda = 10, chi = 1, psi = 1, func = "1/x")
```

hist-methods *Histogram for univariate generalized hyperbolic distributions*

Description

The function hist computes a histogram of the given data values and the univariate generalized hyperbolic distribution.

Usage

```
## S4 method for signature 'ghyp'
hist(x, data = ghyp.data(x), gaussian = TRUE,
     log.hist = F, ylim = NULL, ghyp.col = 1, ghyp.lwd = 1,
     ghyp.lty = "solid", col = 1, nclass = 30, plot.legend = TRUE,
     location = if (log.hist) "bottom" else "topright", legend.cex = 1, ...)
```

Arguments

x	Usually a fitted univariate generalized hyperbolic distribution of class mle.ghyp. Alternatively an object of class ghyp and a data vector.
data	An object coercible to a vector.
gaussian	If TRUE the probability density of the normal distribution is plotted as a reference.
log.hist	If TRUE the logarithm of the histogramm is plotted.
ylim	The "y" limits of the plot.
ghyp.col	The color of the density of the generalized hyperbolic distribution.
ghyp.lwd	The line width of the density of the generalized hyperbolic distribution.
ghyp.lty	The line type of the density of the generalized hyperbolic distribution.
col	The color of the histogramm.
nclass	A single number giving the number of cells for the histogramm.
plot.legend	If TRUE a legend is drawn.
location	The location of the legend. See legend for possible values.
legend.cex	The character expansion of the legend.
...	Arguments passed to plot and qqghyp.

Value

No value is returned.

Author(s)

David Luethi

See Also

qqghyp, fit.ghypuv, hist, legend, plot, lines.

Examples

```
data(smi.stocks)
univariate.fit <- fit.ghypuv(data = smi.stocks[,"SMI"],
                             opt.pars = c(mu = FALSE, sigma = FALSE),
                             symmetric = TRUE)
hist(univariate.fit)
```

---

indices *Monthly returns of five indices*

Description

Monthly returns of indices representing five asset/investment classes *Bonds*, *Stocks*, *Commodities*, *Emerging Markets* and *High Yield Bonds*.

Usage

```
data(indices)
```

Format

`hy.bond` JPMorgan High Yield Bond A (Yahoo symbol "OHYAX").

`emerging.mkt` Morgan Stanley Emerging Markets Fund Inc. (Yahoo symbol "MSF").

`commodity` Dow Jones-AIG Commodity Index (Yahoo symbol "DJI").

`bond` Barclays Global Investors Bond Index (Yahoo symbol "WFBIX").

`stock` Vanguard Total Stock Mkt Idx (Yahoo symbol "VTSMX").

See Also

smi.stocks

Examples

```
data(indices)

pairs(indices)
```

`lik.ratio.test` *Likelihood-ratio test*

Description

This function performs a likelihood-ratio test on fitted generalized hyperbolic distribution objects of class `mle.ghyp`.

Usage

```
lik.ratio.test(x, x.subclass, conf.level = 0.95)
```

Arguments

`x`	An object of class `mle.ghyp`.
`x.subclass`	An object of class `mle.ghyp` whose parameters form a subset of those of `x`.
`conf.level`	Confidence level of the test.

Details

The likelihood-ratio test can be used to check whether a special case of the generalized hyperbolic distribution is the “true” underlying distribution.

The likelihood-ratio is defined as

$$\Lambda = \frac{sup\{L(\theta|\mathbf{X}) : \theta \in \Theta_0\}}{sup\{L(\theta|\mathbf{X}) : \theta \in \Theta\}}.$$

Where L denotes the likelihood function with respect to the parameter θ and data $\mathbf{X}$, and Θ_0 is a subset of the parameter space Θ. The null hypothesis `H0` states that $\theta \in \Theta_0$. Under the null hypothesis and under certain regularity conditions it can be shown that $-2\log(\Lambda)$ is asymtotically chi-squared distributed with ν degrees of freedom. ν is the number of free parameters specified by Θ minus the number of free parameters specified by Θ_0.

The null hypothesis is rejected if $-2\log(\Lambda)$ exceeds the `conf.level`-quantile of the chi-squared distribution with ν degrees of freedom.

Value

A list with components:

`statistic`	The value of the L-statistic.
`p.value`	The p-value for the test.
`df`	The degrees of freedom for the L-statistic.
`H0`	A boolean stating whether the null hypothesis is `TRUE` or `FALSE`.

Author(s)

David Luethi

References

Linear Statistical Inference and Its Applications by C. R. Rao
Wiley, New York, 1973

See Also

fit.ghypuv, logLik, AIC and stepAIC.ghyp.

Examples

```
data(smi.stocks)

sample <- smi.stocks[, "SMI"]

t.symmetric <- fit.tuv(sample, silent = TRUE, symmetric = TRUE)
t.asymmetric <- fit.tuv(sample, silent = TRUE)

# Test symmetric Student-t against asymmetric Student-t in case
# of SMI log-returns
lik.ratio.test(t.asymmetric, t.symmetric, conf.level = 0.95)
# -> keep the null hypothesis

set.seed(1000)
sample <- rghyp(1000, student.t(gamma = 0.1))

t.symmetric <- fit.tuv(sample, silent = TRUE, symmetric = TRUE)
t.asymmetric <- fit.tuv(sample, silent = TRUE)

# Test symmetric Student-t against asymmetric Student-t in case of
# data simulated according to a slightly skewed Student-t distribution
lik.ratio.test(t.asymmetric, t.symmetric, conf.level = 0.95)
# -> reject the null hypothesis

t.symmetric <- fit.tuv(sample, silent = TRUE, symmetric = TRUE)
ghyp.asymmetric <- fit.ghypuv(sample, silent = TRUE)

# Test symmetric Student-t against asymmetric generalized
# hyperbolic using the same data as in the example above
lik.ratio.test(ghyp.asymmetric, t.symmetric, conf.level = 0.95)
# -> keep the null hypothesis
```

---

logLik-AIC-methods *Extract Log-Likelihood and Akaike's Information Criterion*

Description

The functions `logLik` and `AIC` extract the Log-Likelihood and the Akaike's Information Criterion from fitted generalized hyperbolic distribution objects. The Akaike information criterion is calculated according to the formula $-2 \cdot \text{log-likelihood} + k \cdot n_{par}$, where n_{par} represents the number of parameters in the fitted model, and $k = 2$ for the usual AIC.

Usage

```
## S4 method for signature 'mle.ghyp'
logLik(object, ...)

## S4 method for signature 'mle.ghyp'
AIC(object, ..., k = 2)
```

Arguments

`object`	An object of class `mle.ghyp`.
`k`	The "penalty" per parameter to be used; the default k = 2 is the classical AIC.
`...`	An arbitrary number of objects of class `mle.ghyp`.

Value

Either the Log-Likelihood or the Akaike's Information Criterion.

Note

The Log-Likelihood as well as the Akaike's Information Criterion can be obtained from the function `ghyp.fit.info`. However, the benefit of `logLik` and `AIC` is that these functions allow a call with an arbitrary number of objects and are better known because they are generic.

Author(s)

David Luethi

See Also

`fit.ghypuv`, `fit.ghypmv`, `lik.ratio.test`, `ghyp.fit.info`, `mle.ghyp-class`

Examples

```
data(smi.stocks)

## Multivariate fit
fit.mv <- fit.hypmv(smi.stocks, nit = 10)
AIC(fit.mv)
logLik(fit.mv)

## Univariate fit
fit.uv <- fit.tuv(smi.stocks[, "CS"], control = list(maxit = 10))
AIC(fit.uv)
logLik(fit.uv)

# Both together
AIC(fit.uv, fit.mv)
logLik(fit.uv, fit.mv)
```

`mean-vcov-skew-kurt-methods`

Expected value, variance-covariance, skewness and kurtosis of generalized hyperbolic distributions

Description

The function `mean` returns the expected value. The function `vcov` returns the variance in the univariate case and the variance-covariance matrix in the multivariate case. The functions `ghyp.skewness` and `ghyp.kurtosis` only work for univariate generalized hyperbolic distributions.

Usage

```
## S4 method for signature 'ghyp'
mean(x)

## S4 method for signature 'ghyp'
vcov(object)

ghyp.skewness(object)

ghyp.kurtosis(object)
```

Arguments

`x, object`	An object inheriting from class `ghyp`.

Details

The functions `ghyp.skewness` and `ghyp.kurtosis` are based on the function `ghyp.moment`. Numerical integration will be used in case a Student.t or variance gamma distribution is submitted.

Value

Either the expected value, variance, skewness or kurtosis.

Author(s)

David Luethi

See Also

`ghyp`, `ghyp-class`, `Egig` to compute the expected value and the variance of the generalized inverse gaussian mixing distribution distributed and its special cases.

Examples

```
## Univariate: Parametric
vg.dist <- VG(lambda = 1.1, mu = 10, sigma = 10, gamma = 2)
mean(vg.dist)
vcov(vg.dist)
ghyp.skewness(vg.dist)
ghyp.kurtosis(vg.dist)

## Univariate: Empirical
vg.sim <- rghyp(10000, vg.dist)
mean(vg.sim)
var(vg.sim)

## Multivariate: Parametric
vg.dist <- VG(lambda = 0.1, mu = c(55, 33), sigma = diag(c(22, 888)), gamma = 1:2)
mean(vg.dist)
vcov(vg.dist)

## Multivariate: Empirical
vg.sim <- rghyp(50000, vg.dist)
colMeans(vg.sim)
var(vg.sim)
```

`pairs-methods` *Pairs plot for multivariate generalized hyperbolic distributions*

Description

This function is intended to be used as a graphical diagnostic tool for fitted multivariate generalized hyperbolic distributions. An array of graphics is created and qq-plots are drawn into the diagonal part of the graphics array. The upper part of the graphics matrix shows scatter plots whereas the lower part shows 2-dimensional histogramms.

Usage

```
## S4 method for signature 'ghyp'
pairs(x, data = ghyp.data(x), main = "'ghyp' pairwise plot",
      nbins = 30, qq = TRUE, gaussian = TRUE,
      hist.col = c("white", topo.colors(40)),
      spline.points = 150, root.tol = .Machine$double.eps^0.5,
      rel.tol = root.tol, abs.tol = root.tol^1.5, ...)
```

Arguments

`x`	Usually a fitted multivariate generalized hyperbolic distribution of class `mle.ghyp`. Alternatively an object of class `ghyp` and a data matrix.
`data`	An object coercible to a `matrix`.
`main`	The title of the plot.

nbins	The number of bins passed to hist2d.
qq	If TRUE qq-plots are drawn.
gaussian	If TRUE qq-plots with the normal distribution are plotted.
hist.col	A vector of colors passed to hist2d.
spline.points	The number of support points when computing the quantiles used by the qq-plot. Passed to qqghyp.
root.tol	The tolerance of the quantiles. Passed to uniroot via qqghyp.
rel.tol	The tolerance of the quantiles. Passed to integrate via qqghyp.
abs.tol	The tolerance of the quantiles. Passed to integrate via qqghyp.
...	Arguments passed to plot and axis.

Author(s)

David Luethi

See Also

pairs, fit.ghypmv, qqghyp, hist2d

Examples

```
data(smi.stocks)
fitted.smi.stocks <- fit.NIGmv(data = smi.stocks[1:200, ])
pairs(fitted.smi.stocks)
```

---

plot-lines-methods *Plot univariate generalized hyperbolic densities*

Description

These functions plot probability densities of generalized hyperbolic distribution objects.

Usage

```
## S4 method for signature 'ghyp,missing'
plot(x, range = qghyp(c(0.001, 0.999), x), length = 1000, ...)
## S4 method for signature 'ghyp'
lines(x, range = qghyp(c(0.001, 0.999), x), length = 1000, ...)
```

Arguments

x	An univariate ghyp object.
range	The range over which the density will be plotted. The default is the range from the 0.1 % quantile to the 99.9 % quantile. When range has a length greater than 2 it is assumed to be the vector of quantiles and the density is computed on range.
length	The desired length of the density vector.
...	Arguments passed to plot and lines respectively.

Details

When the density is very skewed, the computation of the quantile may fail. See qghyp for details.

Author(s)

David Luethi

See Also

hist, qqghyp, pairs, plot, lines.

Examples

```
data(smi.stocks)

smi.fit   <-  fit.tuv(data = smi.stocks[,"SMI"], symmetric = TRUE)
nestle.fit <- fit.tuv(data = smi.stocks[,"Nestle"], symmetric = TRUE)

## Student-t distribution
plot(smi.fit, type = "l", log = "y")
lines(nestle.fit, col = "blue")

## Empirical
lines(density(smi.stocks[,"SMI"]), lty = "dashed")
lines(density(smi.stocks[,"Nestle"]), lty = "dashed", col = "blue")
```

---

portfolio.optimize *Portfolio optimization with respect to alternative risk measures*

Description

This function performs a optimization of a portfolio with respect to one of the risk measures “sd”, “value.at.risk” or “expected.shortfall”. The optimization task is either to find the *global minimum risk* portfolio, the *tangency* portfolio or the *minimum risk* portfolio given a target-return.

Usage

```
portfolio.optimize(object,
                   risk.measure = c("sd", "value.at.risk", "expected.shortfall"),
                     type = c("minimum.risk", "tangency", "target.return"),
                     level = 0.95, distr = c("loss", "return"),
                     target.return = NULL, risk.free = NULL,
                     silent = FALSE, ...)
```

Arguments

`object`	A multivariate ghyp object representing the loss distribution. In case `object` gives the return distribution set the argument `distr` to "return".
`risk.measure`	How risk shall be measured. Must be one of "sd" (standard deviation), "value.at.risk" or "expected.shortfall".
`type`	The tpye of the optimization problem. Must be one of "minimum.risk", "tangency" or "target.return" (see **Details**).
`level`	The confidence level which shall be used if `risk.measure` is either "value.at.risk" or "expected.shortfall".
`distr`	The default distribution is "loss". If `object` gives the return distribution set `distr` to "return".
`target.return`	A numeric scalar specifying the target return if the optimization problem is of `type` "target.return".
`risk.free`	A numeric scalar giving the risk free rate in case the optimization problem is of `type` "tangency".
`silent`	If `TRUE` no prompts will appear in the console.
`...`	Arguments passed to `optim`.

Details

If `type` is "minimum.risk" the global minimum risk portfolio is returned.

If `type` is "tangency" the portfolio maximizing the slope of "(expected return - risk free rate) / risk" will be returned.

If `type` is "target.return" the portfolio with expected return `target.return` which minimizes the risk will be returned.

Note that in case of an elliptical distribution (symmetric generalized hyperbolic distributions) it does not matter which risk measure is used. That is, minimizing the standard deviation results in a portfolio which also minimizes the value-at-risk et cetera.

Value

A list with components:

`portfolio.dist`	An univariate generalized hyperbolic object of class ghyp which represents the distribution of the optimal portfolio.
`risk.measure`	The risk measure which was used.
`risk`	The risk.
`opt.weights`	The optimal weights.
`converged`	Convergence returned from `optim`.
`message`	A possible error message returned from `optim`.
`n.iter`	The number of iterations returned from `optim`.

Note

In case `object` denotes a non-elliptical distribution and the risk measure is either "value.at.risk" or "expected.shortfall", then the type "tangency" optimization problem is not supported.

Constraints like avoiding short-selling are not supported yet.

Author(s)

David Luethi

See Also

`transform`, `fit.ghypmv`

Examples

```
data(indices)

t.object <- fit.tmv(-indices, silent = TRUE)
gauss.object <- fit.gaussmv(-indices)

t.ptf <- portfolio.optimize(t.object,
                            risk.measure = "expected.shortfall",
                            type = "minimum.risk",
                            level = 0.99,
                            distr = "loss",
                            silent = TRUE)

gauss.ptf <- portfolio.optimize(gauss.object,
                                risk.measure = "expected.shortfall",
                                type = "minimum.risk",
                                level = 0.99,
                                distr = "loss")

par(mfrow = c(1, 3))

plot(c(t.ptf$risk, gauss.ptf$risk),
     c(-mean(t.ptf$portfolio.dist), -mean(gauss.ptf$portfolio.dist)),
     xlim = c(0, 0.035), ylim = c(0, 0.004),
     col = c("black", "red"), lwd = 4,
     xlab = "99 percent expected shortfall",
     ylab = "Expected portfolio return",
     main = "Global minimum risk portfolios")

legend("bottomleft", legend = c("Asymmetric t", "Gaussian"),
       col = c("black", "red"), lty = 1)

plot(t.ptf$portfolio.dist, type = "l",
     xlab = "log-loss ((-1) * log-return)", ylab = "Density")
```

```
lines(gauss.ptf$portfolio.dist, col = "red")

weights <- cbind(Asymmetric.t = t.ptf$opt.weights,
                 Gaussian = gauss.ptf$opt.weights)

barplot(weights, beside = TRUE, ylab = "Weights")
```

---

qq-ghyp *Quantile-Quantile Plot*

Description

This function is intended to be used as a graphical diagnostic tool for fitted univariate generalized hyperbolic distributions. Optionally a qq-plot of the normal distribution can be added.

Usage

```
qqghyp(object, data = ghyp.data(object), gaussian = TRUE, line = TRUE,
       main = "Generalized Hyperbolic Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       ghyp.pch = 1, gauss.pch = 6, ghyp.lty = "solid",
       gauss.lty = "dashed", ghyp.col = "black", gauss.col = "black",
       plot.legend = TRUE, location = "topleft", legend.cex = 0.8,
       spline.points = 150, root.tol = .Machine$double.eps^0.5,
       rel.tol = root.tol, abs.tol = root.tol^1.5, add = FALSE, ...)
```

Arguments

`object`	Usually a fitted univariate generalized hyperbolic distribution of class `mle.ghyp`. Alternatively an object of class `ghyp` and a data vector.
`data`	An object coercible to a `vector`.
`gaussian`	If `TRUE` a qq-plot of the normal distribution is plotted as a reference.
`line`	If `TRUE` a line is fitted and drawn.
`main`	An overall title for the plot.
`xlab`	A title for the x axis.
`ylab`	A title for the y axis.
`ghyp.pch`	A plotting character, i.e., symbol to use for quantiles of the generalized hyperbolic distribution.
`gauss.pch`	A plotting character, i.e., symbol to use for quantiles of the normal distribution.
`ghyp.lty`	The line type of the fitted line to the quantiles of the generalized hyperbolic distribution.
`gauss.lty`	The line type of the fitted line to the quantiles of the normal distribution.
`ghyp.col`	A color of the quantiles of the generalized hyperbolic distribution.

gauss.col	A color of the quantiles of the normal distribution.
plot.legend	If TRUE a legend is drawn.
location	The location of the legend. See legend for possible values.
legend.cex	The character expansion of the legend.
spline.points	The number of support points when computing the quantiles. Passed to qghyp.
root.tol	The tolerance of the quantiles. Passed to uniroot.
rel.tol	The tolerance of the quantiles. Passed to integrate.
abs.tol	The tolerance of the quantiles. Passed to integrate.
add	If TRUE the points are added to an existing plot window. The legend argument then becomes deactivated.
...	Arguments passed to plot.

Author(s)

David Luethi

See Also

hist, fit.ghypuv, qghyp, plot, lines

Examples

```
data(smi.stocks)

smi <- fit.ghypuv(data = smi.stocks[, "Swiss.Re"])

qqghyp(smi, spline.points = 100)

qqghyp(fit.tuv(smi.stocks[, "Swiss.Re"], symmetric = TRUE),
       add = TRUE, ghyp.col = "red", line = FALSE)
```

---

scale-methods *Scaling and Centering of ghyp Objects*

Description

scale centers and/or scales a generalized hyperbolic distribution to zero expectation and/or unit variance.

Usage

```
## S4 method for signature 'ghyp'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

`x`	An object inheriting from class `ghyp`.
`center`	A logical value stating whether the object shall be centered to zero expectation.
`scale`	A logical value stating whether the object shall be scaled to unit variance.

Value

An object of class `ghyp`.

Author(s)

David Luethi

See Also

`transform`, `mean`, `vcov`.

Examples

```
data(indices)

t.fit <- fit.tmv(indices)
gauss.fit <- fit.gaussmv(indices)

## Compare the fitted Student-t and Gaussian density.
par(mfrow = c(1, 2))

## Once on the real scale...
plot(t.fit[1], type = "l")
lines(gauss.fit[1], col = "red")

## ...and once scaled to expectation = 0, variance = 1
plot(scale(t.fit)[1], type = "l")
lines(scale(gauss.fit)[1], col = "red")
```

---

smi.stocks — *Daily returns of five swiss blue chips and the SMI*

Description

Daily returns from January 2000 to January 2007 of five swiss blue chips and the Swiss Market Index (SMI).

Usage

```
data(smi.stocks)
```

Format

`SMI` Swiss Market Index.

`Novartis` Novartis pharma.

`CS` Credit Suisse.

`Nestle` Nestle.

`Swisscom` Swiss telecom company.

`Swiss.Re` Swiss reinsurer.

See Also

`indices`

Examples

```
data(smi.stocks)

pairs(smi.stocks)
```

`stepAIC.ghyp` *Perform a model selection based on the AIC*

Description

This function performs a model selection in the scope of the generalized hyperbolic distribution class based on the Akaike information criterion. `stepAIC.ghyp` can be used for the univariate as well as for the multivariate case.

Usage

```
stepAIC.ghyp(data, dist = c("ghyp", "hyp", "NIG", "VG", "t", "gauss"),
             symmetric = NULL, ...)
```

Arguments

`data`	A `vector`, `matrix` or `data.frame`.
`dist`	A character vector of distributions from where the best fit will be identified.
`symmetric`	Either `NULL`, `TRUE` or `FALSE`. `NULL` means that both symmetric and asymmetric models will be fitted. For symmetric models select `TRUE` and for asymmetric models select `FALSE`.
`...`	Arguments passed to `fit.ghypuv` or `fit.ghypmv`.

Value

A list with components:

`best.model`	The model minimizing the AIC.
`all.models`	All fitted models.
`fit.table`	A `data.frame` with columns `model`, `symmetric`, `lambda`, `alpha.bar`, `aic`, `llh` (log-Likelihood), `converged`, `n.iter` (number of iterations) sorted according to the aic. In the univariate case three additional columns containing the parameters `mu`, `sigma` and `gamma` are added.

Author(s)

David Luethi

See Also

`lik.ratio.test`, `fit.ghypuv` and `fit.ghypmv`.

Examples

```
data(indices)

# Multivariate case:
aic.mv <- stepAIC.ghyp(indices, dist = c("ghyp", "hyp", "t", "gauss"),
                       symmetric = NULL, control = list(maxit = 500),
                       silent = TRUE, nit = 500)

summary(aic.mv$best.model)

# Univariate case:
aic.uv <- stepAIC.ghyp(indices[, "stock"], dist = c("ghyp", "NIG", "VG", "gauss"),
                       symmetric = TRUE, control = list(maxit = 500), silent = TRUE)


# Test whether the ghyp-model provides a significant improvement with
# respect to the VG-model:
lik.ratio.test(aic.uv$all.models[[1]], aic.uv$all.models[[3]])
```

`summary-method` *mle.ghyp summary*

Description

Produces a formatted output of a fitted generalized hyperbolic distribution.

Usage

```
## S4 method for signature 'mle.ghyp'
summary(object)
```

Arguments

object An object of class mle.ghyp.

Value

Nothing is returned.

Author(s)

David Luethi

See Also

Fitting functions fit.ghypuv and fit.ghypmv, coef, mean, vcov and ghyp.fit.info for accessor functions for mle.ghyp objects.

Examples

```
data(smi.stocks)
mle.ghyp.object <- fit.NIGmv(smi.stocks[, c("Nestle", "Swiss.Re", "Novartis")])
summary(mle.ghyp.object)
```

---

transform-extract-methods

Linear transformation and extraction of generalized hyperbolic distributions

---

Description

The transform function can be used to linearly transform generalized hyperbolic distribution objects (see **Details**). The extraction operator [extracts some margins of a multivariate generalized hyperbolic distribution object.

Usage

```
## S4 method for signature 'ghyp'
transform(`_data`, summand, multiplier)

## S3 method for class 'ghyp'
x[i = c(1, 2)]
```

Arguments

`_data`	An object inheriting from class `ghyp`.
`summand`	A `vector`.
`multiplier`	A `vector` or a `matrix`.
`x`	A multivariate generalized hyperbolic distribution inheriting from class `ghyp`.
`i`	Index specifying which dimensions to extract.
`...`	Arguments passed to `transform`.

Details

If $X \sim GH$, `transform` gives the distribution object of "multiplier * X + summand", where X is the argument named `_data`.

If the object is of class `mle.ghyp`, iformation concerning the fitting procedure (cf. `ghyp.fit.info`) will be lost as the return value is an object of class `ghyp`.

Value

An object of class `ghyp`.

Author(s)

David Luethi

See Also

`scale`, `ghyp`, `fit.ghypuv` and `fit.ghypmv` for constructors of `ghyp` objects.

Examples

```
## Mutivariate generalized hyperbolic distribution
multivariate.ghyp <- ghyp(sigma=var(matrix(rnorm(9),ncol=3)), mu=1:3, gamma=-2:0)

## Dimension reduces to 2
transform(multivariate.ghyp, multiplier=matrix(1:6,nrow=2), summand=10:11)

## Dimension reduces to 1
transform(multivariate.ghyp, multiplier=1:3)

## Simple transformation
transform(multivariate.ghyp, summand=100:102)

## Extract some dimension
multivariate.ghyp[1]
multivariate.ghyp[c(1, 3)]
```

Index