Package ‘GEVStableGarch’

August 3, 2015

Type Package

Title ARMA-GARCH/APARCH models with GEV and stable distributions

Version 1.1

Date 2015-07-19

Author Thiago do Rego Sousa, Cira Etheowalda Guevara Otiniano and Silvia Regina Costa Lopes

Maintainer Thiago do Rego Sousa <thiagoestatistico@gmail.com>

Description Package for simulation and estimation of ARMA-GARCH/APARCH models with GEV and stable distributions.

License GPL (>=2)

Depends R(>= 2.15.0),fGarch,fExtremes,stabledist,skewt,Rsolnp

Additional_repositories http://www.robustanalysis.com

R topics documented:

GEVStableGarch-package *ARMA-GARCH/APARCH modelling with GEV and stable distributions*

Description

This package is designed to perform maximum likelihood estimation of ARMA-GARCH/APARCH models with GEV, stable and generalized asymmetric t (GAt). Other common conditional distributions such as the normal, skew Student's t and GED are also allowed.

Details

Package:	GEVStableGarch
Type:	Package
Version:	1.1
Date:	2015-07-19
License:	GPL(>=2)
Depends:	R(>= 2.15.0), fGarch, fExtremes, stabledist, skewt, Rsolnp

GARCH models have proven to be highly effective for analyzing financial data over the past decades. In particular, the combination of ARMA-GARCH models with stable and generalized extreme distribution (GEV) distributions was successfully applied for forecasting volatility and for the measurement of Value at Risk (VAR).

Choosing the normal distribution as probability distribution for the innovations was a common choice in the beginning of the development of ARCH-type models. But recent research Nolan (1999), Mittnik et al. (2002), Mittnik and Paolella (2003), Curto et al. (2006), Frain (2009), Zhao et al. (2011) has shown that other distributions should be considered, specially because normal distribution can not account for fat tails and asymmetry found in real data.

This package contains functions for estimating and simulating combined ARMA-GARCH or ARMA-APARCH models with error distributions following GEV or stable densities. Our implementation also allows the estimation using maximum log-likelihood under different assumptions, Normal, Student's t, skew Student's t from Fernandez and Steel (1998), Generalized Asymmetric t (GAt) from Paolella (1997) and Generalized Error Distribution (GED).

The current version of package **GEVStableGarch** has a new algorithm that allows the user to enforce stationarity during estimation. Aditionally, it contains functions for selecting the best model according to a goodness-of-fit criteria.

Time Series Simulation

contains functions to simulate ARMA-GARCH/APARCH processes with conditional GEV or stable distributions. Note: These routines were adapted from /codegarchSpec and /codegarchSim functions available in **fGarch** package.

Functions:

`gsSpec`	Specifies an univariate ARMA-GARCH/APARCH model,
`gsSim`	Simulates a ARMA-GARCH/APARCH process.

Parameter Estimation

contains functions to fit the parameters of ARMA-GARCH/APARCH time series processes.

Functions:

`gsFit`	Fits the parameters of a GARCH process. This function also provides an algorithm to enforce stationarity du
`gsSelect`	Selects the best model according to a goodness-of-fit criteria.

Other Conditional Distribution Functions

contains functions to compute density, distribution, quantiles and generate random values using important conditional distributions used in the garch literature.

Functions:

`[dpqr]skstd`	Skew Student's t distribution function from Fernandez and Steel (1998),
`[dpqr]gat`	Generalized Asymmetric t distribution (GAt) from Paolella (1997). Also known as t3-distribution,
`gsMomentAparch`	Computes APARCH Moments for the following distributions: stable, GEV, GAt, normal, skew Stude

Author(s)

Thiago do Rego Sousa, Cira Etheowalda Guevara Otiniano and Silvia Regina Costa Lopes.

Maintainer: Thiago do Rego Sousa <thiagoestatistico@gmail.com>

`gat` *Generalized Asymmetric t Distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the generalized asymmetric t distribution (GAt). Notice that this is the same t3-distribution mentioned in the literature and defined by Paolella (1997). The GAt distribution includes the Student's t, Laplace, Cauchy and the normal distribution when the shape parameter (

$$\nu \to \infty$$

) (see see Mittnik and Paolella (2000)).

Usage

```
dgat(x, mean = 0, sd = 1, nu = 2, d = 3, xi = 1, log = FALSE)
pgat(q, mean = 0, sd = 1, nu = 2, d = 3, xi = 1)
qgat(p, mean = 0, sd = 1, nu = 2, d = 3, xi = 1)
rgat(n, mean = 0, sd = 1, nu = 2, d = 3, xi = 1)
```

Arguments

`mean, sd, d, nu, xi`

location parameter `mean`, scale parameter `sd`, shape 1 parameter `nu`, shape 2 parameter `d`, asymmetry parameter `xi`.

`n` the number of observations.

`p` a numeric vector of probabilities.

`x, q` a numeric vector of quantiles.

`log` a logical; if TRUE, densities are given as log densities.

Value

`d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates,
all values are numeric vectors.

Author(s)

Thiago do Rego Sousa

References

Mittnik, S., Paolella, M.S. (2000). *Prediction of Financial Downside-Risk with Heavy-Tailed Conditional Distributions* Available at SSRN 391261.

Paolella, M. (1997). *Tail Estimation and Conditional Modeling of Heteroskedstic Time-Series.* Ph.D Thesis, Institute of Statistics and Econometrics, Christian Albrechts University of Kiel.

Examples

```
# Simulate Random Values and compare with
# the empirical density and probability functions
# Note: This example was addapted from "sstd {fGarch} R Documentation"

# Configure plot and generate random values
par(mfrow = c(2, 2))
set.seed(1000)
r = rgat(n = 1000)
plot(r, type = "l", main = "GAt Random Values", col = "steelblue")

# Plot empirical density and compare with true density:
hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
box()
x = seq(min(r), max(r), length = 201)
lines(x, dgat(x), lwd = 2)

# Plot density function and compare with true df:
plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
lines(x, pgat(x), lwd = 2)

# Compute quantiles:
# Here we compute the quantiles corresponding to the probability points from
# -10 to 10 and expect to obtain the same input sequence
round(qgat(pgat(q = seq(-10, 10, by = 0.5))), digits = 6)
```

`GEVSTABLEGARCH-class` *Class* `"GEVSTABLEGARCH"`

Description

The class GEVSTABLEGARCH represents an ARMA-GARCH/APARCH model estimated wih function `gsFit`

Objects from the Class

Objects can be created by calling function `gsFit`. This object contain the estimated parameters of a time series process.

Slots

`call`**:** Object of class `"call"`: the call of the `gsFit` function.

`formula`**:** Object of class `"formula"`: a formula object specifying mean and variance equation.

`method`**:** Object of class `"character"`: a string describing the optimization method used to search for the optimum value

`convergence`**:** Object of class `"numeric"`: an integer code. 0 indicates successful convergence of of the estimation method used to perform the optimization of the log-likelihood function. A value different from zero indicates a failure in achieving convergence. Notice that sometimes the optimization algorithm will return a "true" convergence, even when the optimized negative log-likelihood equals to 1e99. In this case, we set the variable `convergence` to 1 to indicate that convergence was not achieved. Aditional interpretation of `convergence` codes can be made by using the R help of the corresponding optimization routine: `?solnp` for the `"sqp"` or `"sqp.restriction} algorithms and \code{?nlminb} for \code{"nlminb"} or \code{"nlminb+nm` algorithms.

`messages`**:** Object of class `"list"`: a character string giving additional informations collected during estimation.

`data`**:** Object of class `"numeric"`: a numeric vector containing the data of the time series to be estimated.

`fit`**:** Object of class `"list"`: a list with the results from the parameter estimation.

`residuals`**:** Object of class `"numeric"`: a numeric vector with the residual values.

`h.t`**:** Object of class `"numeric"`: a numeric vector with the conditional variances.

`sigma.t`**:** Object of class `"numeric"`: a numeric vector with the conditional standard deviations.

`title`**:** Object of class `"character"`: a string with the title.

`description`**:** Object of class `"character"`: a string with a description.

Methods

show `signature(object = "GEVSTABLEGARCH")`: prints an object of class 'GEVSTABLEGARCH'.

Author(s)

Thiago do Rego Sousa

References

Brockwell, P.J., Davis, R.A. (1996). *Introduction to Time Series and Forecasting*. Springer, New York.

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

`GEVSTABLEGARCHSPEC-class`

Class `"GEVSTABLEGARCHSPEC"`

Description

This class represents an ARMA-GARCH/APARCH model with innovations following an i.i.d stable, GEV distributions. Other distributions are also allowed.

Objects from the Class

Objects can be created by calls of the function `gsSpec`. This object specifies an univariate ARMA-GARCH/APARCH model with GEV or stable conditional distribution. Note that we only allow the user to create an object that do not violate the model specifications. For more details about the definition of those models see: see Mittnik et al. (2002) (stable innovations), Zhao et al. (2011) (GEV innovations) Wuertz et al. (2008) (finite variance innovations) and Mittnik et al. (2003) (GAt innovations).

Slots

`call`**:** Object of class `"call"`: the call of the `gsSpec` function.

`formula`**:** Object of class `"formula"`: a list with two formula entries for the mean and variance equation of the combined ARMA-GARCH/APARCH model.

`model`**:** Object of class `"list"`: a list with the model parameters specifying the Autorregressive, the GARCH/APARCH and the innovations parameters.

`presample`**:** Object of class `"matrix"`: a numeric matrix with presample values.

`distribution`**:** Object of class `"character"`: a character string with the name of the conditional distribution.

`rseed`**:** Object of class `"numeric"`: an integer with the random number generator seed.

Methods

show `signature(object = "GEVSTABLEGARCHSPEC")`: prints an object of class 'GEVSTABLEGARCH'

Author(s)

Thiago do Rego Sousa.

References

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

Wuertz, D., Chalabi, Y., Luksan, L. (2009). *Parameter Estimation of ARMA Models with GARCH/APARCH Errors: An R and SPlus SoftwareImplementation*. Journal of Statistical Software, forthcoming, http://www-stat.wharton.upenn.edu/~steele/...WurtzEtAlGarch.pdf.

`gsFit` *Estimation of ARMA-GARCH/APARCH models*

Description

This function uses Maximum Likelihood technique to estimate the parameters of ARMA-GARCH or ARMA-APARCH model with several conditional distributions. The user can also enforce stationarity during estimation

Usage

```
gsFit(formula = ~garch(1, 1), data, cond.dist = c("stableS0", "stableS1",
"stableS2", "gev", "gat", "norm", "std", "sstd", "skstd", "ged"),
include.mean = TRUE, algorithm = c("sqp", "sqp.restriction",
"nlminb", "nlminb+nm"), control = NULL,
tolerance = NULL, title = NULL, description = NULL)
```

Arguments

`formula` a formula object specifying mean and variance equation. The mean equation should be specified using the names: "arma"", "garch"" or "aparch". For example: ~arma(1,0)-garch(1,0) for AR(1)-ARCH(1), and ~arma(0,1)-garch(2,2) for MA(1)-GARCH(2,2).

`data` a numeric vector containing the data of the time series to be estimated.

`cond.dist` a character string naming conditional distribution of innovations. The package was created to accept the following distributions: `"stableS0"` (stable in S0-parameterization), `"stableS1"` (stable in S1-parameterization), `"stableS2"` (stable in S2-parameterization), `"gev"` and `"gat"`. Other common distributions are also possible such as the `"norm"`, `"std"`, `"sstd"` (skew Student's t defined in **fGarch** package), `"skstd"` (skew Student's t from Fernandez and Steel (1998)), `"ged"`.

`include.mean` This is a boolean variable. It intercept is TRUE than we estimate the model with intercept.

`algorithm` The algorithm to be used to search for the optimum value. The current version of the **GEVStableGarch** package implements four different optimization procedures, namely the `"sqp"`, `"sqp.restriction"` (enforcing stationarity) , `"nlminb"` and `"nlminb+nm"`. See the details for more information about the implemented algorithms.

`control` Control parameters used either in “SQP” routine (see the RSolnp documentation) or in “nlminb” (see the nlminb R documentation).

`tolerance` Tolerance for parameter estimation. When specified, the tolerance parameter should be passed as a list in the following format `list( TOLG = 1e-8, TOLSTABLE = 1e-2, TOLSTA` where `TOLG` is the tolerance used to set the boundary of the general model, `TOLSTABLE` is the boundary of parameters for stable distribution and `TOLSTATIONARITY` is the tolerance parameter used by the "sqp.restriction"" algorithm when searching for the stationary solution.

`title` a string with the title.

`description` a string with a description.

Details

The starting values are key in getting any possible convergence value and they were chosen to reflect this. For example, the GEV shape starting value was chosen to be 0.01 since in many cases we can not obtain good results. There is also the possibility to fit models with conditional “Normal”, “t-student” and “skew t-student” distributions but the main purpose of this routine is to deal with “GEV” and “stable”. The parameters will be interpreted according to the following equations (see Wurtz et al. ,2009)

$$X_t = \mu + \sum_{i=1}^{m} a_i X_{t-i} + \sum_{j=1}^{n} b_j \varepsilon_{t-j} + \varepsilon_t$$

$$\varepsilon_t = \sigma_t z_t, \quad z_t \overset{iid}{\sim} D(0,1) \ ,$$

$$\sigma_t^\delta = \omega + \sum_{i=1}^{p} \alpha_i (\varepsilon_{t-i} - \gamma_i |\varepsilon_{t-i}|)^\delta + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^\delta$$

where

$$\mathcal{D}_\vartheta(0,1)$$

is the density of the innovations with zero location and unit scale and

$$\vartheta$$

are additional distributional parameters that describe the skew and the shape of the distribution.

ESTIMATION ALGORITHM Most software packages implement the estimation of GARCH models without imposing sta- tionarity, but restricting the parameter set by appropriate bounds. This last approach was implemented in the GEVStableGarch package through the following algorithms: "sqp", "nlminb" and "nlminb + nm". The first two algorithms search for the optimum value by restricting the parameter set to appropriate lower and upper bounds. The last implements a two step optimization procedure, which consists in starting the search by using the con- strained routine nlminb and then performing another search using an unconstrained method (in our case the Nelder-Mead method implemented in the R base function optim).

Most software packages implement the estimation of GARCH models without imposing stationarity, but restricting the parameter set by appropriate bounds. This last approach was implemented in the **GEVStableGarch** package through the following algorithms: `"sqp"`, `"nlminb"` and `"nlminb + nm"`. The first two algorithms search for the optimum value by restricting the parameter set to appropriate lower and upper bounds. The last implements a two step optimization procedure, which consists in starting the search by using the constrained routine `nlminb` and then performing another search using an unconstrained method (in our case the Nelder-Mead method implemented in the `R` base function `optim`). This approach was suggested by Wuertz et al. (2009) since in many cases it leads to an improved solution (in terms of the likelihood function of the data).

Finally, the "sqp.restriction" algorithm gives us the possibility to maximize the log-likelihood function in such a way that the estimated model is still stationary. What made this possible was the availability of the APARCH moments formulas throughout the literature for many conditional distributions and the derivation of the stable case given in this paper.

Value

gsFit
returns a S4 object of class "GEVSTABLEGARCH" with the following slots:

@call	the call of the gsFit function.
@formula	a list with two formula entries, one for the mean and the other one for the variance equation.
@method	a string denoting the optimization method.
@data	a numeric vector containing the data of the estimated time sereis.
@fit	a list with the results from the parameter estimation: par - the estimated parameters; llh - the estimated negative log-likelihood function; hessian - the hessian matrix returned by the optimization algorithm; ics - the value of the goodness-of-fit measures (AIC, AICc and BIC) (See Brockwell and Davis, 2002 for more details); order - a list with the ARMA and garch ORDERS; cond.dis - the conditional distribution; se.coef - standard errors of the estimated parameters; tValue - tValue of the estimated parameters; matcoef - an organized matrix with the estimated parameters.
@residuals	a numeric vector with the residual values (ε_t).
@h.t	a numeric vector with the residual values ($h_t = \sigma_t^\delta$).
@sigma.t	a numeric vector with the conditional standard deviation σ_t.
@title	a string with the title.
@description	a string with a description.

The entries of the @fit slot show the results from the optimization.

Enhances

The estimation of ARMA-GARCH/APARCH models with conditional stable distribution is mainly dependent on the time taken during the calculation of density points. Although the package can be used with the R **stabledist** package, the estimation of such models is only feasible if we use the fast implementation of stable densities using the R **stable** package designed by Nolan. The **stable** package implements a faster computation of stable densities that are accurately enough to perform numerical optimization. Package **stable** is available at http://www.robustanalysis.com.

Author(s)

Thiago do Rego Sousa, Cira Etheowalda Guevara Otiniano and Silvia Regina Costa Lopes

References

Brockwell, P.J., Davis, R.A. (1996). *Introduction to Time Series and Forecasting*. Springer, New York.

Mittnik, S., Paolella, M.S., Rachev, S.T. (2002). *Stationarity of stable power-GARCH processes*. Journal of Econometrics 106, 97–107.

Nolan, J.P. (1997). *Numerical calculations of stable densities and distribution functions*. Communications in Statistics - Stochastic Models 13, 759–774.

Nolan, J.P. (1997). *Maximum likelihood estimation and diagnostics for stable distribution*. In O. E.Barndorff-Nielsen, T. Mikosch, e S. I. Resnick (Eds.), Levy Processes: Theory andApplications, Boston, Birkhauser, 379–400.

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

Wuertz, D., Chalabi, Y., Luksan, L. (2009). *Parameter Estimation of ARMA Models with GARCH/ APARCH Errors: An R and SPlus SoftwareImplementation*. Journal of Statistical Software, forthcoming, http://www-stat.wharton.upenn.edu/~steele/...WurtzEtAlGarch.pdf.

Zhao, X., Scarrott, C.J., Oxley, L., Reale, M. (2011). *GARCH dependence in extreme value models with Bayesian inference*. Mathematics an Computers in Simulation, 81, Issue 7, 1430–1440.

Yinyu, Ye. (1987). *Interior Algorithms for Linear, Quadratic, and Linearly Constrained Non-Linear Programming*. Ph.D. Thesis, Department of EES, Stanford University.

See Also

`gsSelect`

Examples

```
# This examples uses the dataset of the package fGarch to estimate
# an ARMA(1,1)-GARCH(1,1) with GEV conditional distribution.
library(fGarch)
data(dem2gbp)
x = dem2gbp[, 1]
gev.model = gsFit(data = x , formula = ~garch(1,1), cond.dist = "norm")
```

`gsMomentAparch` *Computation of moments for several conditional distribution*

Description

Computation of the moments expression

$$E(|Z| - \gamma Z)^{\delta}$$

where

$$\mathcal{D}_{\vartheta}(0,1)$$

is the density of the innovations with zero location and unit scale and

$$\vartheta$$

are additional distributional parameters that describe the skew and the shape of the distribution.

Usage

```
gsMomentAparch(cond.dist = c("stableS1", "gev", "gat", "norm",
"std", "sstd", "skstd", "ged"), shape = 1.5, skew = 0, delta = 1, gm = 0)
```

Arguments

`cond.dist` a character string naming conditional distribution of innovations. The package was created to accept the following distributions: `"stableS0"`, `"stableS1"`, `"stableS2"`, `"gev"` and `"gat"`. Other common distributions are also possible such as the `"norm"`, `"std"`, `"sstd"`, `"skstd"`, `"ged"`.

`shape, skew` The shape and skew parameter for the conditional distribution. For the stable distribution the shape and skew are the

$$\alpha$$

(index of stability) and

$$\beta$$

(asymmetry) parameters. For the Generalized Asymmetric t distribution the shape parameters

$$\nu$$

and

$$d$$

and

$$\xi$$

is the skew parameter. For the skew Student's t distribution the shape is the degrees of freedom

$$\nu$$

and

$$\xi$$

is the asymmetry parameter. Finally, for the GEV distribution the shape parameter is the

$$\xi$$

parameter.

`delta, gm` The

$$\delta$$

and

$$\gamma$$

parameteres of the ARMA-GARCH/APARCH model.

See the R documentation of function `gsFit` for more details.

Details

For each conditional distribution, the evaluation of the moment expression

$$E(|Z| - \gamma Z)^\delta$$

is crucial if one is interested in finding a stationary solution. Analytical expressions to compute these moments are available throughout the GARCH literature for many conditional distributions and we have derived an efficient formula for the asymmetric stable case. the APARCH moment expression, which is not valid for the `sstd` density. It is true that we can use numerical integration to compute those moments but there are several drawbacks that represent limitations. The first one is that numerical integration is always slower than using an closed expression to compute those formulas. The second is that this technique fails when the distribution becomes very pick or very concentrated on a small portion of the real line. For example, if we use the R base function `integrate` to calculate the area total probability on the interval

$$(-\infty, \infty)$$

for the standard gat distribution with parameters

$$\nu = 2, d = 0.05$$

and

$$\xi = 1$$

we get

$$3.690067e - 07$$

, which is not true since the total probability must be 1. Therefore, it is very important for us to have the exact expression for all those moments in order to evaluate correctly the stationarity restrictions.

The `gsMomentAparch` function can be used to compute those expressions for the following distributions: Normal, Student's t, skewed Student's t from Fernandez and Steel (1998), GED, stable in its 1-parameterization and the GAt distribution. Only for the GEV distribution we use numerical integration. For all other distribution we use the analytical expression to compute the APARCH moment.

Note: The **GEVStableGarch** package implements a slighty different version of the Student's skew distribution that is different from the one implemented inside the **fGarch** package. Therefore, the APARCH moment formula is only valid for the **GEVStableGarch** implementation (`skstd`) and not for the **fGarch** implementation `sstd`.

Value

Returns the following expression for several conditional distributions

$$E(|Z| - \gamma Z)^{\delta}$$

. If any of the input parameters are outside the parameter space, this function returns

$$\infty$$

Author(s)

Thiago do Rego Sousa.

References

Ding, Z., Granger, C., Engle, R.F. (1993). *A Long Memory Property of Stock Market Returns and a New Model.* Journal of Empirical Finance, 1, 83–106.

Diongue, A.K. (2008). *An investigation of Stable-Paretian Asymmetric Power GARCH Model.* Journal des Sciences, 8(4), 15–26.

Lambert, P., Laurent, S. (2001). *Modelling Financial Time Series Using GARCH-type Models with a Skewed Student Distribution for the Innovations.* Institut de Statistique, Universite Catholique de Louvain, Discussion Paper 0125.

Mittnik, S., Paolella, M.S. (2000). *Conditional Density and Value-At-Risk Prediction of Asian Currency Exchange Rates.* Journal of Forecasting, 19(4), 313–333.

Mittnik, S., Paolella, M.S., Rachev, S.T. (2002). *Stationarity of stable power-GARCH processes.* Journal of Econometrics 106, 97–107.

Examples

```
# Computation of the Moment E( |Z| - gamma Z) ^ delta for several distributions

gsMomentAparch(cond.dist = "stableS1", shape = 1.1, skew = 0, delta = 1.01, gm = 0.99999)

gsMomentAparch(cond.dist = "gev", shape = -4, skew = 0, delta = 1.4, gm = 0)

gsMomentAparch(cond.dist = "gat", shape = c(1.9,2.3), skew = 0.5, delta = 0.4, gm = 0)

gsMomentAparch(cond.dist = "norm", shape = c(1.9,2.3), skew =1, delta = 11.4, gm = -0.999)

gsMomentAparch(cond.dist = "std", shape = 2.001, skew = -0.5, delta = 2, gm = -0.99)

gsMomentAparch(cond.dist = "sstd", shape = 2.001, skew = 0.11, delta = 2, gm = -0.99)

gsMomentAparch(cond.dist = "skstd", shape = 5.001, skew = 0.11, delta = 3, gm = -0.5)

gsMomentAparch(cond.dist = "ged", shape = 6, skew = 0.11, delta = 5.11, gm = -0.5)
```

---

gsSelect *Selects the best model according to goodness-of-fit criteria*

Description

This function estimates ARMA-GARCH/APARCH models with varying order and returns the smallest goodness-of-fit criteria

Usage

```
gsSelect(data, order.max = c(1, 1, 1, 1),
selection.criteria = c("AIC", "AICc", "BIC"), is.aparch = FALSE,
cond.dist = c("stableS0", "stableS1", "stableS2", "gev",
"GAt", "norm", "std", "sstd", "skstd", "ged"),
include.mean = TRUE,
algorithm = c("sqp", "sqp.restriction", "nlminb", "nlminb+nm"), ...)
```

Arguments

`data` Data with the time series to be estimated. This object can not contain NULL elements.

`order.max` Maximum order of models to search. It must by a vector of the type `c(mMax, nMax, pMax, qMax)` with each entry representing the maximum order of the model that will be fitted when searching for the best model.

`selection.criteria` The goodness-of-fit measure to use be minimized. Three different criterias are allowed: AIC, AICc or BIC. For more information see Brockwell and Davis (1996)

`is.aparch` Boolean variable indicating whether to search for ARMA-GARCH or ARMA-APARCH models `cond.dist` a character string naming conditional distribution of innovations. The package was created to accept the following distributions: `"stableS0"`, `"stableS1"`, `"stableS2"`, `"gev"` and `"GAt"`. Other common distributions are also possible such as the `"norm"`, `"std"`, `"sstd"`, `"skstd"`, `"ged"`.

`include.mean` This is a boolean variable. It intercept is TRUE than we estimate the model with intercept.

`algorithm` The algorithm to be used to search for the optimum value. The current version of the **GEVStableGarch** package implements four different optimization procedures, namely the `"sqp"`, `"sqp.restriction"` (enforcing stationarity) , `"nlminb"` and `"nlminb+nm"`. See the details for more information about the implemented algorithms.

`...` Additional parameters to be passed to function `gsFit` during estimation.

Value

Returns a S4 object of class "GEVSTABLEGARCH" with the best model. See the Value section of `gsFit` function for details.

Author(s)

Thiago do Rego Sousa.

References

Brockwell, P.J., Davis, R.A. (1996). *Introduction to Time Series and Forecasting*. Springer, New York.

Examples

```
# Best ARMA-GARCH model within the range ARMA(0,0)-GARCH(1,0) to ARMA(0,0)-GARCH(1,1)
# using the Corrected Akaike Information Criteria (AICc)
library(fGarch)
data(dem2gbp)
x = dem2gbp[,1]

model = gsSelect (data = x, order.max = c(0,0,1,1), is.aparch = FALSE,
          algorithm = "sqp", cond.dist = "gev", selection.criteria = "AIC")
```

`gsSim` *Simulation of ARMA-GARCH/APARCH process*

Description

This functions simulate time series following ARMA-GARCH/APARCH models with GEV and stable distributions.

Usage

```
gsSim(spec = garchSpec(), n = 100, n.start = 100)
```

Arguments

`spec` A model specified with function gsSpec

`n` The size of simulated time series

`n.start` length of the "burn-in" period of the simulated time series

Details

The initial values of the time series are fixed and the recursion formulas of the model are used to simulate the dynamics of the process. We do not verify the stationarity conditions of the model because the simulation of non-stationary process could also be of interest by the researcher.

Value

The function returns an object containing the following items:

`model` A string describing the estimated model.

`cond.dist` The conditional distribution used to fit the model.

`series` An array with three columns, where the first column contains the simulated ARMA-GARCH/APARCH process (`garch`), the second column the conditional standard deviations `"sigma"`, and the last column the innovations named `"eps"`.

Author(s)

Thiago do Rego Sousa.

References

Brockwell, P.J., Davis, R.A. (1996). *Introduction to Time Series and Forecasting*. Springer, New York.

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

Wuertz, D., Chalabi, Y., Luksan, L. (2009). *Parameter Estimation of ARMA Models with GARCH/ APARCH Errors: An R and SPlus SoftwareImplementation*. Journal of Statistical Software, forthcoming, http://www-stat.wharton.upenn.edu/~steele/...WurtzEtAlGarch.pdf.

Examples

```
# Simulation of a ARMA-APARCH process with stable conditional distribution
#x <- GSgarch.Sim(N = 2500, mu = 0.1,a = c(0.2,0.3),b = c(0.2,0.5),
#omega = 0.1, alpha = c(0.1,0.2),beta = c(0.1,0.1),gm=c(0.3,-0.3),
#delta = 1,skew = 0.3,shape = 1.9, cond.dis = "stable")
```

gsSpec *Specification of ARMA-GARCH/APARCH models with GEV or stable distributions*

Description

Specifies an ARMA-GARCH or ARMA-APARCH model with innovations following GEV, stable.

Usage

```
gsSpec(model = list(), presample = NULL,
    cond.dist = c("stableS0", "stableS1", "stableS2", "gev",
    "gat", "norm", "std", "sstd", "skstd", "ged"),
    rseed = NULL)
```

Arguments

`model` a list of ARMA-GARCH/APARCH model parameters:
`omega` - the coefficient of the variance equation;
`alpha` - the value or vector of autoregressive coefficients;
`beta` - the value or vector of variance coefficients;
The values for the ARMA part are:
`mu` - the mean value;
`ar` - the autoregressive coefficients;
`ma` - the moving average coefficients.
The parameters for the conditional distributions are:
`skew` - the skewness parameter;
`shape` - the shape parameter.

`presample` presample - a numeric "matrix" with 3 columns and with max(m,n,p,q) rows. The first culumn are the innovations, the second the conditional variances, and the last the time series. When presample is missing, it is constructed as [z,h,y] where z ~ Normal(0,1), h = "uev" recursion initialization described in Wuertz et al. (2009) and y = mu. Note that the conditional variance column can contain only strictly positive numbers and the function `gsSpec` check for invalid values.

`cond.dist` a character string naming conditional distribution of innovations. The package was created to accept the following distributions: `"stableS0"`, `"stableS1"`, `"stableS2"`, `"gev"` and `"gat"`. Other common distributions are also possible such as the `"norm"`, `"std"`, `"sstd"`, `"skstd"`, `"ged"`, but they are not the main feature of this work.

`rseed` the seed for the intitialization of the random number generator for the innovations.

Details

This functions uses the interface of the `garchSpec` routine from package **fGarch** to simulate random values of the ARMA-GARCH/APARCH model with conditional GEV or stable distribution.

Value

The returned value is an object of class `"GEVSTABLEGARCHSPEC"`.

Author(s)

Thiago do Rego Sousa.

References

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

Wuertz, D., Chalabi, Y., Luksan, L. (2009). *Parameter Estimation of ARMA Models with GARCH/ APARCH Errors: An R and SPlus SoftwareImplementation*. Journal of Statistical Software, forthcoming, http://www-stat.wharton.upenn.edu/~steele/...WurtzEtAlGarch.pdf.

Examples

```
# stable-GARCH from Curto et al. (2009) for the DJIA dataset
spec.stable = gsSpec(model = list(mu = 0.0596, omega = 0.0061,
alpha = 0.0497, beta = 0.9325, skew = -0.9516, shape = 1.9252),
cond.dist = "stableS1")
sim.stable = gsSim(spec = spec.stable, n = 1000)

# GEV-GARCH model from Zhao et al. (2011)
spec.gev = gsSpec(model = list(mu = 0.21, a = 0.32, omega = 0.01,
alpha = 0.45, beta = 0.08, shape = 0.08), cond.dist = "gev")
sim.gev = gsSim(spec = spec.gev, n = 1000)
```

`show-methods` *GEVSTABLEGARCH Package Show Methods*

Description

Methods to organize the output to the user when printing objects of class GEVSTABLEGARCH

Methods

`signature(object = "GEVSTABLEGARCH")` Print function for objects of class `"GEVSTABLEGARCH"`

`signature(object = "GEVSTABLEGARCHSPEC")` Print function for objects of class `"GEVSTABLEGARCHSPEC"`

Author(s)

Thiago do Rego Sousa.

References

Wuertz, D., Chalabi, Y., with contribution from Miklovic, M., Boudt, C., Chausse, P., and others (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82*, `http://CRAN.R-project.org/package=fGarch`.

sksd *Skew Student's t Distribtuion from Fernandez and Steel (1997)*

Description

Functions to compute density, distribution function, quantile function and to generate random values for the Skew Student's t distribtuion from Fernandez and Steel (1997). Notice that this function is different from the skew Student's t `sstd` from package fGarch. Although the two distributions use the same approach from Fernandez and Steel (1997), the second one was reparameterized in such a way that when the parameters mean = 0 and sd = 1 the distribution will have a true mean equal zero and the true variance equals to one, no matter the value of the skew parameter. The distributions `skstd` and `sstd` are the same when the asymmetry parameter `xi` equals to 1, in which case they correspond to the well known Student's t distribution `std`.

Usage

```
dskstd(x, mean = 0, sd = 1, nu = 3, xi = 1, log = FALSE)
pskstd(q, mean = 0, sd = 1, nu = 3, xi = 1)
qskstd(p, mean = 0, sd = 1, nu = 3, xi = 1)
rskstd(n, mean = 0, sd = 1, nu = 3, xi = 1)
```

Arguments

`mean, sd, nu, xi`
location parameter `mean`, scale parameter `sd`, shape parameter `nu`, asymmetry parameter `xi`.

`n` the number of observations.

`p` a numeric vector of probabilities.

`x, q` a numeric vector of quantiles.

`log` a logical; if TRUE, densities are given as log densities.

Value

`d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates.
All values are numeric vectors.

Author(s)

Thiago do Rego Sousa.

References

Fernandez, C., Steel, M.F. (1998). *On Bayesian Modeling of Fat Tails and Skewness*. Journal of the American Statistical Association, Taylor & Francis Group, 93(441), 359–371.

Examples

```
# Simulate Random Values and compare with
# the empirical density and probability functions
# Note: This example was addapted from "sstd {fGarch} R Documentation"

# Configure plot and generate random values
par(mfrow = c(2, 2))
set.seed(1000)
r = rskstd(n = 1000)
plot(r, type = "l", main = "Skew Student's t Random Values", col = "steelblue")

# Plot empirical density and compare with true density:
hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
box()
x = seq(min(r), max(r), length = 201)
lines(x, dskstd(x), lwd = 2)

# Plot density function and compare with true df:
plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
lines(x, pskstd(x), lwd = 2)

# Compute quantiles:
# Here we compute the quantiles corresponding to the probability points from
# -10 to 10 and expect to obtain the same input sequence
round(qskstd(pskstd(q = seq(-10, 10, by = 0.5))), digits = 6)
```

Index