Package 'GAS'

October 30, 2026

Type Package

Title Generalized Autoregressive Score Models

Version 0.4.3

Description Simulation, estimation and forecasting of univariate and multivariate generalized autoregressive score models. Time-varying distributional parameters are updated using the score of the conditional distribution, with alternative score scaling specifications. Provides maximum likelihood estimation, filtering, density forecasting and rolling forecast evaluation for a range of conditional distributions. Includes diagnostic tools and methods for evaluating Value-at-Risk and Expected Shortfall forecasts. The models and their applications are described in Ardia et al. (2019) <doi:10.18637/jss.v088.i06>.

License GPL-3

BugReports https://github.com/LeopoldoCatania/GAS/issues

URL https://github.com/LeopoldoCatania/GAS

LazyData TRUE

Imports Rcpp (>= 0.12.2), Rsolnp, MASS, xts, numDeriv, zoo, cubature

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.4.0), methods

Suggests testthat

NeedsCompilation yes

Author Leopoldo Catania [aut, cre] (ORCID: <https://orcid.org/0000-0002-0981-1921>), David Ardia [ctb] (ORCID: <https://orcid.org/0000-0003-2823-782X>), Kris Boudt [ctb] (ORCID: <https://orcid.org/0000-0002-1000-5142>)

Maintainer Leopoldo Catania <leopoldo.catania@econ.au.dk>

Repository CRAN

Date/Publication 2026-10-30 15:00:02 UTC

Contents

GAS-package . . . 3
BacktestDensity . . . 4
BacktestVaR . . . 6
ConfidenceBands . . . 7
cpichg . . . 10
DistInfo . . . 10
distributions . . . 11
dji30ret . . . 13
fn.optim . . . 14
fn.solnp . . . 15
FZLoss . . . 16
Goals . . . 18
mGASFit . . . 19
mGASFor . . . 20
mGASRoll . . . 21
mGASSim . . . 22
mGASSpec . . . 23
MultiGASFit . . . 23
MultiGASFor . . . 26
MultiGASRoll . . . 28
MultiGASSim . . . 30
MultiGASSpec . . . 32
MultiMapParameters . . . 34
MultiUnmapParameters . . . 35
NumericalBounds . . . 37
PIT_test . . . 38
plot-methods . . . 40
sp500ret . . . 41
sp500rv . . . 42
StockIndices . . . 42
tqdata . . . 43
uGASFit . . . 44
uGASFor . . . 45
uGASRoll . . . 46
uGASSim . . . 48
uGASSpec . . . 49
UniGASFit . . . 49
UniGASFor . . . 53
UniGASRoll . . . 55
UniGASSim . . . 57
UniGASSpec . . . 59
UniMapParameters . . . 60
UniUnmapParameters . . . 62
usunp . . . 63

Index **64**

GAS-package *Generalized Autoregressive Score models in R*

Description

The GAS package allows us to simulate, estimate and forecast using univariate and multivariate Generalized Autoregressive Score (GAS) models (also known as Dynamic Conditional Score (DCS) models), see e.g., Creal et. al. (2013) and Harvey (2013). A detailed implementation of the package functionalities are reported in Ardia et. al. (2018, 2019).

Details

The authors acknowledge Google for financial support via the Google Summer of Code 2016 project "GAS"; see https://summerofcode.withgoogle.com/archive/2016/projects/4537082387103744/.

Note

By using GAS you agree to the following rules:

- You must cite Ardia et al. (2019) in working papers and published papers that use GAS. Use `citation("GAS")`.
- You must place the following URL in a footnote to help others find GAS: https://CRAN.R-project.org/package=GAS.
- You assume all risk for the use of GAS.

Author(s)

Leopoldo Catania [aut,cre], Kris Boudt [ctb], David Ardia [ctb]

Maintainer: Leopoldo Catania <leopoldo.catania@econ.au.dk>

References

Ardia D, Boudt K and Catania L (2018). "Downside Risk Evaluation with the R Package GAS." R Journal, 10(2), 410-421. doi:10.32614/RJ2018064.

Ardia D, Boudt K and Catania L (2019). "Generalized Autoregressive Score Models in R: The GAS Package." Journal of Statistical Software, 88(6), 1-28. doi:10.18637/jss.v088.i06.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

`BacktestDensity` *Backtest a series of one-step ahead density predictions.*

Description

The `BacktestDensity()` function accepts an object of the class uGASRoll, and returns a `list` with two elements: (i) the averages Negative Log Score (NLS) and weighted Continuous Ranked Probability Score (wCRPS) introduced by Gneiting and Ranjan (2011), and (ii) their values at each point in time. The wCRPS is evaluated using 5 weight functions, see Details.

Usage

```
BacktestDensity(Roll, lower, upper, K = 1000, a = NULL, b = NULL)
```

Arguments

`Roll`	an object of the class uGASRoll.
`lower`	`numeric` the lower bound used to approximate the wCRSP by deterministic numerical integration as detailed in Gneiting and Ranjan (2011). This coincides with y_l in Equation 16 of Gneiting and Ranjan (2011).
`upper`	`numeric` the upper bound used to approximate the wCRSP by deterministic numerical integration as detailed in Gneiting and Ranjan (2011). This coincides with y_u in Equation 16 of Gneiting and Ranjan (2011).
`K`	`numeric` integer representing the number of points used to discretize the wCRPS integral. This is I in Equation 16 of Gneiting and Ranjan (2011). By default `K = 1000`.
`a`	`numeric`. mean of the normal distribution used in the weight function. By default `a = NULL`, which means that it is set equal to the empircial mean of the in sample observations.
`b`	`numeric`. standard deviation of the normal distribution used in the weight function. By default `b = NULL`, which means that it is set equal to the empircial standard deviation of the in sample observations.

Details

The integral is approximated on [lower,upper] by a left-endpoint Riemann sum with K points and spacing (upper-lower)/K. K must be a positive integer, bounds must be finite with lower less than upper, and b must be finite and strictly positive. The finite integration range truncates the full CRPS integral. The tail weight equals $1-\exp(-\frac{1}{2}((z-a)/b)^2)$ and is nonnegative. The returned columns are ordered as uniform, center, tails, tail_r and tail_l.

The average Negative Log Score (NLS) is computed as the negative of the average of the log scores evaluated during the out-of-sample period. The average weighted Continuous Ranked Probability Score (wCRPS) is computed as the average of the wCRPS evaluated during the out-of-sample period, see Gneiting and Ranjan (2011).
The wCRPS is evaluated using Equation 16 of Gneiting and Ranjan (2011). The weights functions implemented are:

- $w(z) = 1$: Uniform,
- $w(z) = \phi_{a,b}(z)$: Center,
- $w(z) = 1 - \phi_{a,b}(z)/\phi_{a,b}(a)$: Tails,
- $w(z) = \Phi_{a,b}(z)$: Right tail,
- $w(z) = 1 - \Phi_{a,b}(z)$: Left tail,

where $\phi_{a,b}(z)$ and $\Phi_{a,b}(z)$ are the pdf and cdf of a Gaussian distribution with mean a and standard deviation b, respectively. The label "Uniform" represents the case where equal emphasis is given to all the parts of the distribution.

Value

A `list` with elements: `average`, `series`. The element "`average`" is a named vector with the averages NLS and wCRSP. The element "`series`" is a `list`: the fist element, `LS`, contains the out-of-sample Log Score (not with the negative sign), the second element, `WCRPS`, contains a `matrix` with the wCRPS series. The columns of this matrix are named: "`uniform`", "`center`", "`tails`", "`tail_r`", "`tail_l`", which are associated with the wCRSP with emphasis on: Uniform, Center, Tails, Right tail and Left tail, respectively.

Author(s)

Leopoldo Catania

References

Gneiting T, Ranjan R (2011). "Comparing Density Forecasts using Threshold -and Quantile-Weighted Scoring Rules." Journal of Business & Economic Statistics, 29(3), 411-422. doi:10.1198/jbes.2010.08110.

Examples

```
data("cpichg")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE, scale = TRUE,
                                   shape = FALSE))

Roll = UniGASRoll(cpichg, GASSpec, ForecastLength = 50,
                  RefitEvery = 10, RefitWindow = c("moving"))

BackTest = BacktestDensity(Roll, lower = -100, upper = 100)

BackTest$average
```

BacktestVaR *Backtest Value at Risk (VaR)*

Description

This function implements several backtesting procedures for the Value at Risk (VaR). These are: (i) The statistical tests of Kupiec (1995), Christoffesen (1998) and Engle and Manganelli (2004), (ii) The tick loss function detailed in Gonzalez-Rivera et al. (2004), the mean and max absolute loss used by McAleer and Da Veiga (2008) and the actual over expected exceedance ratio.

Usage

```
BacktestVaR(data, VaR, alpha, Lags = 4)
```

Arguments

data	numeric Vector of observations.
VaR	numeric Vector containing the VaR series.
alpha	numeric The VaR confidence level.
Lags	numeric Lags used in the Dynamic Quantile test of Engle and Manganelli (2004).

Details

This function implements several backtesting procedure for the Value at Risk. The implemented statistical tests are:

- LRuc The unconditional coverage test of Kupiec (1995).
- LRcc The conditional coverage test of Christoffesen (1998).
- DQ The Dynamic Quantile test of Engle and Manganelli (2004).

The implemented VaR backtesting quantities are:

- AD mean and maximum absolute deviation between the observations and the quantiles as in McAleer and Da Veiga (2008).
- Loss Average quantile loss and quantile loss series as in Gonzalez-Rivera et al. (2004).
- AE Actual over Expected exceedance ratio.

Value

A list with elements: LRuc, LRcc, DQ, AD, AE.

Author(s)

Leopoldo Catania

References

Christoffersen PF (1998). "Evaluating Interval Rorecasts." International Economic Review, 39(4), 841-862.

Engle RF and Manganelli S. (2004). "CAViaR: Conditional Autoregressive Value at Risk by Regression Quantiles." Journal of Business & Economic Statistics, 22(4), 367-381. doi:10.1198/073500104000000370.

Gonzalez-Rivera G, Lee TH, and Mishra, S (2004). "Forecasting Volatility: A Reality Check Based on Option Pricing, Utility Function, Value-at-Risk, and Predictive Likelihood." International Journal of Forecasting, 20(4), 629-645. doi:10.1016/j.ijforecast.2003.10.003.

Kupiec PH (1995). "Techniques for Verifying the Accuracy of Risk Measurement Models." The Journal of Derivatives, 3(2), 73-84. doi:10.3905/jod.1995.407942

McAleer M and Da Veiga B (2008). "Forecasting Value-at-Risk with a Parsimonious Portfolio Spillover GARCH (PS-GARCH) Model." Journal of Forecasting, 27(1), 1-19. doi:10.1002/for.1049.

Examples

```
data("StockIndices")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = FALSE, scale = TRUE,
                                   shape = FALSE))

FTSEMIB = StockIndices[, "FTSEMIB"]

InSampleData  = FTSEMIB[1:500]
OutSampleData = FTSEMIB[501:1000]

# Parameter standard errors are not needed for this example.
Fit = UniGASFit(GASSpec, InSampleData, Compute.SE = FALSE)

Forecast = UniGASFor(Fit, Roll = TRUE, out = OutSampleData)

alpha = 0.05

VaR = quantile(Forecast, alpha)

BackTest = BacktestVaR(OutSampleData, VaR, alpha)
```

`ConfidenceBands` *Pointwise confidence bands for time-varying parameters*

Description

Construct pointwise confidence bands for time-varying distribution parameters by sampling model coefficients from their estimated asymptotic normal distribution and filtering the observed data for each draw.

Usage

```
ConfidenceBands(object, B = 10000L,
                probs = c(0.01, 0.1, 0.9, 0.99), ...)
```

Arguments

`object`	A fitted object of class `uGASFit` or `mGASFit`.
`B`	A positive integer giving the number of coefficient draws.
`probs`	A nonempty numeric vector of probabilities between zero and one. The returned quantiles follow the supplied order.
`...`	Currently unused.

Details

The function implements the in-sample simulation-based approach described in Section 3.3 of Blasques et al. (2016).

Model coefficients are sampled on the unrestricted estimation scale. The covariance estimate is the inverse of the summed outer product of per-observation likelihood scores, stored as `optimiser$opg` in the fitted object. Reliable use of the normal approximation requires nonsingular information at an interior optimum. The function rejects fits failing the convergence or gradient checks and rejects invalid or numerically singular OPG information matrices. No generalized inverse or automatic positive-definiteness repair is used. Refit older objects, or objects fitted with `Compute.SE = FALSE`, with `Compute.SE = TRUE` before calling this function.

For a fit using conditional OPG inference, only the retained coefficients are sampled, using the inverse of their principal OPG submatrix. Excluded coefficients stay fixed at their fitted values in every draw. A warning marks these bands as conditional on those coefficients. The retained OPG must pass the usual checks; the singular full matrix is not inverted.

Each sampled coefficient vector is transformed using the mapping appropriate to the fitted model. For multivariate models, the fitted `ScalarParameters` specification is respected. The observed data are then filtered under each sampled vector.

Quantiles are computed separately for each time point and each distribution parameter, using the B simulated paths. The fitted path is not included as an additional simulation draw.

These are pointwise parameter confidence bands. They are not simultaneous bands over the full path or prediction intervals for future observations.

For a central 95 percent pointwise interval, use `probs = c(0.025, 0.975)`.

Value

A numeric array with dimensions `c(T + 1L, length(probs), K)`, where T is the number of observations and K is the number of distribution parameters.

The first dimension indexes time. Its first T entries correspond to the in-sample parameter path, and its final entry corresponds to the one-step-ahead parameter prediction.

The second dimension indexes the requested quantile probabilities. The third dimension indexes the distribution parameters, in the order returned by `getFilteredParameters`.

The array contains quantiles only. The fitted parameter path is available separately through `getFilteredParameters(object`

Author(s)

Leopoldo Catania

References

Blasques F, Koopman SJ, Lasak K, and Lucas, A (2016). "In-sample Confidence Bands and Out-of-Sample Forecast Bands for Time-Varying Parameters in Observation-Driven Models." International Journal of Forecasting, 32(3), 875-887. doi:10.1016/j.ijforecast.2016.04.002.

Examples

```
library("GAS")

data("cpichg")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE, scale = TRUE,
                                   shape = FALSE))

Fit = UniGASFit(GASSpec, cpichg)

Bands <- ConfidenceBands(
    Fit,
    B = 1000L,
    probs = c(0.025, 0.975)
)

lower <- Bands[, 1L, ]
upper <- Bands[, 2L, ]

# Fitted path, returned separately:
fitted_path <- getFilteredParameters(Fit)
```

cpichg — *Data: Quarterly logarithmic change in percentage points of the Consumer Price Index for All Urban Consumers: All Items (CPIAUCSL) from 1947-04-01 to 2016-05-01*

Description

Quarterly logarithmic change in percentage points of the Consumer Price Index for All Urban Consumers: All Items (CPIAUCSL) from 1947-04-01 to 2016-05-01 available at https://fred.stlouisfed.org/series/CPIAUCSL.

Usage

```
data("cpichg")
```

Format

A xts object containing 276 observations from 1947-04-01 to 2016-05-01.

References

US. Bureau of Labor Statistics, Consumer Price Index for All Urban Consumers: All Items [CPIAUCSL], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/CPIAUCSL, June 24, 2016.

DistInfo — *Information for the supported distributions*

Description

Print the information regarding distributions supported in the GAS package.

Usage

```
DistInfo(DistLabel = NULL, N = 2, FULL = TRUE)
```

Arguments

DistLabel	character indicating the label of the conditional distribution. if DistLabel = NULL all the supported distributions are printed. Default DistLabel = NULL

. Run DistLabels() to see the labels of the currently implemented distributions.

N	numeric Indicating the number of asset if DistLabel link to a multivariate distribution. Default N = 2.
FULL	logical If TRUE the function prints all the the information. Default FULL = TRUE

Details

The information are printed in the console.

Value

`DistInfo` prints distribution information to the console and returns `NULL` invisibly.

The other functions documented on this page return:

`DistLabels` A character vector of supported distribution labels.

`DistName` A character string containing the distribution name.

`DistType` A character string: `"univariate"` or `"multivariate"`.

`DistParameters` A character vector naming the distribution's parameter groups.

`DistScalingType` A character string listing the supported score scaling methods, separated by commas.

`NumberParameters` A numeric scalar giving the number of distribution parameters, accounting for the dimension of a multivariate distribution.

Author(s)

Leopoldo Catania

Examples

```
# show the information of all the supported distributions
library("GAS")

DistInfo()
```

---

distributions *Distributions of the GAS package*

Description

Density, distribution function, quantile function, random generator, moments, scores and information matrix of univariate and multivariate distributions of the GAS package.

Usage

```
ddist_Uni(y, Theta, Dist, log = FALSE)
pdist_Uni(q, Theta, Dist)
qdist_Uni(p, Theta, Dist)
rdist_Uni(Theta, Dist)
mdist_Uni(Theta, Dist)
Score_Uni(y, Theta, Dist)
IM_Uni(Theta, Dist)
Quantiles(mTheta, Dist, vProbs)
```

```
ddist_Multi(y, Theta, Dist, log = FALSE)
rdist_Multi(Theta, N, Dist)
Score_Multi(y, Theta, Dist)
```

Arguments

mTheta	A K by T matrix of distribution parameters, with one time point per column.
vProbs	A nonempty numeric vector of probabilities in [0,1].
y, q	numeric Scalar quantile. For Score_Multi and ddist_Multi, y is a numeric vector.
p	numeric Scalar probability.
Theta	numeric Vector of distribution parameters. The order of the parameters is generally: location, scale, skewness, shape, shape2. When the distribution defined by Dist does not have, say, the shape parameter, this should be simply omitted. See also DistInfo for specific distributions.
Dist	character Label of the conditional distribution, see DistInfo.
log	logical If TRUE, the density value $p(y)$ is given as $log(p(y))$. Dy Default log = FALSE.
N	numeric Integer. cross-sectional dimension for the multivariate case.

Details

Quantiles evaluates every requested probability at every time point and returns a T by length(vProbs) matrix, even when the number of probabilities equals T. It does not interpret equal lengths as one probability per time point.

For ast, ast1 and ald, quantiles are evaluated analytically without a finite search interval. Their zero- and one-probability quantiles are minus and plus infinity.

For snorm and sstd, the two-piece split has probability $1/(1+\xi^2)$. Location and scale are the mean and standard deviation after centering and standardizing the two-piece distribution. The sstd parameterization requires degrees of freedom greater than two. Both distributions support "Identity", "Inv" and "InvSqrt" scaling. Their full expected Fisher information matrices, including cross-parameter entries, are evaluated analytically in the mean, standard deviation, skewness and (for sstd) shape coordinates. The information is transformed to the unconstrained state coordinates before scaling. "InvSqrt" uses the inverse lower Cholesky factor.

For skellam, Theta = c(mu, v) denotes mean and variance, with $v > |\mu|$. The underlying Poisson intensities are $(v+\mu)/2$ and $(v-\mu)/2$. Noninteger observations have zero probability mass. The CDF is evaluated at the floor of its argument. For $0 < p < 1$, the quantile is the smallest integer q satisfying $F(q) \geq p$; the endpoints are minus and plus infinity. The implementation returns NA if its bounded numerical search cannot resolve a quantile within exactly representable integer arithmetic.

Kurtosis is reported as ordinary kurtosis, not excess kurtosis. For std, the mean is undefined when degrees of freedom are at most one; variance is infinite when the mean exists but degrees of freedom are at most two. Skewness requires degrees of freedom greater than three. For std and sstd, kurtosis is infinite when variance is finite but degrees of freedom are at most four. Undefined

moments are reported as NA. These conventions agree with ast and ast1 at the corresponding moment-existence boundaries.

The function mdist_Uni returns a vector with four elements: mean, variance, skewness and kurtosis coefficients. The functions Score_Uni and IM_Uni returns the score and the Fisher information matrix for univariate distributions. The function Score_Multi returns the score for multivariate distributions. See DistInfo for the lists of supported distributions. The scalar distribution helpers are not vectorized; Quantiles evaluates a time-by-probability grid. ddist_Uni and ddist_Multi give the density, pdist_Uni gives the distribution function, qdist_Uni gives the quantile function, and rdist_Uni and rdist_Multi generate random deviates.

Value

1. numeric scalar for: ddist_Uni, pdist_Uni, qdist_Uni, rdist_Uni,
2. numeric vector for: Score_Uni, Score_Multi and rdist_Multi,
3. matrix for IM_Uni and Quantiles.

Author(s)

Leopoldo Catania

Examples

```
# Skew Student-t distribution

# log density
Theta = c("location" = 0, "scales" = 1, "skewness" = 1.2, "shape" = 7)

ddist_Uni(y = 0.5, Theta, "sstd", TRUE)

# probability
pdist_Uni(q = -1.69, Theta, "sstd")

#quantile
qdist_Uni(p = 0.05, Theta, "sstd")

#random generator
rdist_Uni(Theta, "sstd")

#moments
mdist_Uni(Theta, "sstd")
```

---

dji30ret *data: Dow Jones 30 Constituents Closing Value Log Return in percentage points*

Description

This dataset is taken from the rugarch package of Ghalanos (2015). Returns are in percentage points.

Dow Jones 30 Constituents closing value log returns from 1987-03-16 to 2009-02-03 from Yahoo Finance. Note that AIG was replaced by KFT (Kraft Foods) on September 22, 2008. This is not reflected in this data set as that would bring the starting date of the data to 2001.

Usage

```
data("dji30ret")
```

Format

A data.frame containing 5,521x30 observations.

Source

Yahoo Finance

References

Ghalanos A (2015). "rugarch: Univariate GARCH models." https://cran.r-project.org/package=rugarch.

---

fn.optim *A wrapper to the optim function.*

Description

Minimize the negative log likelihood using BFGS, with a deterministic alternative solver when numerical checks fail.

Usage

```
fn.optim(par0, data, GASSpec, FUN, Compute.Hessian = FALSE)
```

Arguments

par0	numeric vector of named model coefficients.
data	numeric vector or matrix of data.
GASSpec	An object of the class uGASSpec or mGASSpec, created via the UniGASSpec and MultiGASSpec functions.
Compute.Hessian	Logical, default FALSE. Whether to compute numerical Hessians for the curvature diagnostic and final output.
FUN	A function to optimize.

Details

Optimization uses the average negative log likelihood to stabilize the numerical scale across sample sizes. The primary BFGS controls are `maxit = 2000L`, `reltol = 1e-11`, and `abstol = -Inf`. The starting point is retained as a candidate. If convergence or gradient diagnostics fail (or curvature diagnostics fail when `Compute.Hessian = TRUE`), `stats::nlminb` and a subsequent BFGS refinement are attempted. The refinement supplies Richardson numerical gradients from `numDeriv::grad`, using `reltol = 1e-14`, to avoid the coarse default finite-difference step in `optim`. The best finite objective is retained; among numerically tied objectives, a candidate that passes both solver convergence and gradient checks is preferred. The reported value and optional Hessian refer to the summed negative log likelihood. With `Compute.Hessian = FALSE`, neither numerical Hessian is computed and the returned `hessian` is `NULL`. Convergence and gradient checks, including gradient-based refinement when needed, remain active. Curvature is used for optimizer diagnostics only; fitted-model standard errors and confidence bands use the outer product of per-observation scores. The additional `method` field identifies the selected candidate.

Value

It returns a named list including the following four elements: i) `pars`: a `numeric` vector where the estimated parameters are stored, ii) `value`: a `numeric` containing the value of the negative log likelihood evaluated at its minumum, iii) `hessian`, a `numeric` matrix containing the Hessian matrix evaluated at the minimum of the negative log likelihood (or `NULL` when not requested or evaluation fails), iv) `convergence` a `numeric` element indicating the selected solver convergence result.

Author(s)

Leopoldo Catania

See Also

`help(optim)`

`fn.solnp` *A wrapper to the solnp function of the* `Rsolnp` *package of Ghalanos and Theussl (2016).*

Description

This function is a wrapper to the solnp function of the `Rsolnp` package of Ghalanos and Theussl (2016).

Usage

```
fn.solnp(par0, data, GASSpec, FUN, Compute.Hessian = FALSE)
```

Arguments

par0	numeric vector of named model coefficients.
data	numeric vector or matrix of data.
GASSpec	An object of the class uGASSpec or mGASSpec, created via the UniGASSpec and MultiGASSpec functions.
Compute.Hessian	Logical, default FALSE. Whether to return SOLNP's internally maintained Hessian approximation. This does not disable its internal quasi-Newton updates. No additional numerical Hessian is computed.
FUN	A function to optimize.

Details

The following control parameters are used: trace = 0, rho = 1, outer.iter = 400, inner.iter = 1800, delta = 1e-08, tol = 1e-08. See the documentation of solnp.

Value

It returns a named list with four elements: i) pars: a numeric vector where the estimated parameters are stored, ii) value: a numeric containing the value of the negative log likelihood evaluated at its minumum, iii) hessian, SOLNP's Hessian approximation when requested, otherwise NULL, and iv) convergence a numeric element indicating the convergence results of solnp.

Author(s)

Leopoldo Catania

References

Alexios Ghalanos and Stefan Theussl (2015). "Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method". R package version 1.16.

See Also

help(solnp)

FZLoss *Fissler and Ziegel (2016) (FZ) joint loss function for Value at Risk and Expected Shortfall.*

Description

This function implements Fissler and Ziegel (2016) (FZ) joint loss function for Value at Risk and Expected Shortfall.

Usage

```
FZLoss(data, VaR, ES, alpha)
```

Arguments

data	numeric Vector of observations.
VaR	numeric Vector containing the VaR series.
ES	numeric Vector containing the ES series.
alpha	numeric The VaR and ES confidence level.

Details

This function implements Fissler and Ziegel (2016) (FZ) joint loss function for Value at Risk and Expected Shortfall. The parameterization used is that of Patton et al. (2017) and is given by:

$\frac{1}{\alpha ES_t^\alpha} I_t^\alpha (y_t - VaR_t^\alpha) + \frac{VaR_t^\alpha}{ES_t^\alpha} + \log -ES_t^\alpha - 1.$

See also Fissler et al. (2015).

Value

A numeric vector containing the joing VaR and ES loss values.

Author(s)

Leopoldo Catania

References

Fissler, T., Ziegel, J.F., (2016). "Higher order elicitability and Osband's principle." The Annals of Statistics 44, 1680-1707.
Fissler, T., Ziegel, J.F., Tilmann, G. (2015). "Expected Shortfall is jointly elicitable with Value at Risk - Implications for backtesting." arXiv preprint arXiv:1507.00244.
Patton, A. J., Ziegel, J.F., Chen, R. (2017). "Dynamic semiparametric models for expected shortfall (and Value-at-Risk)." arXiv preprint arXiv:1707.05108.

Examples

```
data("StockIndices")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = FALSE, scale = TRUE,
                                   shape = FALSE))

FTSEMIB = StockIndices[, "FTSEMIB"]

InSampleData  = FTSEMIB[1:500]
OutSampleData = FTSEMIB[501:1000]

# Parameter standard errors are not needed for this example.
```

```
Fit = UniGASFit(GASSpec, InSampleData, Compute.SE = FALSE)

Forecast = UniGASFor(Fit, Roll = TRUE, out = OutSampleData)

alpha = 0.05

vVaR = quantile(Forecast, alpha)
vES  = ES(Forecast, alpha)

FZ = FZLoss(OutSampleData, vVaR, vES, alpha)
```

`Goals` *data: Goals scored by England against Scotland in international football matches.*

Description

Number of goals scored by England against Scotland in international football matches. This is a 116 x 2 zoo object spanning the period 1872-1987 with a yearly frequency. The first column reports the number of goals scored by England against Scotland. The second column is a dummy variable equal 1 for matches played in England. This data set is taken from the Harvey (1989) pg 524.

Usage

```
data("Goals")
```

Format

A zoo object containing 116 x 2 observations.

Source

Harvey (1989) pg. 524

References

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge university press. https://cran.r-project.org/package=rugarch.

mGASFit *Class for the Multivariate GAS fitted object*

Description

Class for the multivariate GAS fitted object.

Details

The show and summary methods report *Average Filtered Parameters*: the arithmetic average of each mapped distribution parameter in GASDyn$mTheta over the first ModelInfo$iT columns. The final column, containing the next-period forecast, is excluded.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

Data: Object of class list. Contains the user's data.

Estimates: Object of class list. Contains: lParList list of estimated parameters, optimiser object delivered from the optimization function, StaticFit ML estimates for the constant model, Inference inferential results for the estimated parameters.

GASDyn: Object of class list. Contains: the series of filtered dynamic (GASDyn$mTheta) for the time–varying parameters, the series of scaled scores (GASDyn$mInnovation), the series of unrestricted filtered parameters (GASDyn$mTheta_tilde), the series of log densities (GASDyn$vLLK), the log likelihood evaluated at its optimum value (GASDyn$dLLK)

ModelInfo: Object of class list. Contains information about the GAS specification:

- Spec Object of the class uGASSpec containing the GAS specification.
- iT numeric Number of observation.
- elapsedTime Numeric elapsed time in seconds.

Methods

- show signature(object = 'mGASFit'): print object information.
- summary signature(object = 'mGASFit'): Show summary.
- plot signature(x='mGASFit', y='missing'): Plot filtered dynamic and other estimated quantities.
- getFilteredParameters signature(object = "mGASFit"): Extract filtered parameters.
- getObs signature(object = "mGASFit"): Extract original observations.
- coef signature(object = 'uGASFit'): Returns a named vector of estimated coefficients. Also accepts the additional logical argument do.list. If do.list = TRUE, estimated coefficients are organized in a list with arguments: vKappa the intercept vector, mA the A system matrix, mB the B system matrix. By default, do.list = FALSE.

- getMoments `signature(object = "mGASFit")`: Extract conditional moments.
- `residuals` `signature(object = 'mGASFit')`: Extract the residuals. Also accepts the additional logical argument `standardize`. If `standardize = TRUE`, residuals are standardized by cholesky of the filtered covariance matrix. By default `standardize = FALSE`.
- `convergence` `signature(object = 'mGASFit')`: Extract convergence information.

Author(s)

Leopoldo Catania

`mGASFor` *Class for the Multivariate GAS Forecast object*

Description

Class for the multivariate GAS forecast object.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Forecast`: Object of class `list`. Contains forecasts:

- `PointForecast`: `matrix` with parameters forecasts.
- `Moments`: `list` with centered moments forecasts. The first element contains a `matrix` with the predicted conditional means. The second element contains an `array` with the predicted conditional covariances.
- `vLS`: `numeric` Log Score (Predictive Log Likelihood).

`Bands`: `array` with confidence bands parameters forecasts. Available only if `Roll = TRUE`.

`Draws`: If `ReturnsDraws = TRUE` it is a iH x iB `matrix` of draws from the predictive distribution.

`Info`: `list` with forecast information.

`Data`: `list` with original data.

Methods

- `show` `signature(object = "uGASFor")`: Show summary.
- `plot` `signature(x='uGASFor',y='missing')`: Plot forecasted quantities.
- `getForecast` `signature(object = "uGASFor")`: Extract parameters forecast.
- `getObs` `signature(object = "uGASFor")`: Extract original observations.
- `getMoments` `signature(object = "uGASFor")`: Extract moments forecasts.
- `LogScore` `signature(object = "uGASFor")`: Extract Log Scores.

Author(s)

Leopoldo Catania

mGASRoll *Class for the Multivariate GAS Rolling object*

Description

Class for the multivariate GAS rolling object.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Forecast`: Object of class `list`. Contains forecasts:

- `PointForecast`: `matrix` with parameters forecasts.
- `Moments`: `list` with centered moments forecasts. The first element contains a `matrix` with the predicted conditional means. The second element contains an `array` with the predicted conditional covariances.
- `vLS`: `numeric` Log Score (Predictive Log Likelihood).

`Info`: `list` with forecast information.

`Data`: `list` with original data.

Methods

- `show signature(object = 'mGASRoll')`: Show summary.
- `plot signature(x = 'mGASRoll',y = 'missing')`: Plot forecasted quantities.
- `getForecast signature(object = 'mGASRoll')`: Extract parameters forecast.
- `getObs signature(object = 'mGASRoll')`: Extract original observations.
- `getMoments signature(object = 'mGASRoll')`: Extract moments forecasts.
- `LogScore signature(object = 'mGASRoll')`: Extract Log Scores.
- `residuals signature(object = 'mGASRoll')`: Extract the forecast errors. Also accepts the additional logical argument `standardize`. If `standardize = TRUE`, forecast errors are standardized by cholesky of the forecast covariance matrix. By default `standardize = FALSE`.
- `coef signature(object = 'mGASFit')`: Returns a matrix of estimated coefficients. Each row of the matrix corresponds to a refit of the model during the forecast period according to the `RefitEvery` argument provided in the MultiGASRoll function. Also accepts the additional logical argument `do.list`. If `do.list = TRUE`, estimated coefficients are organized in a list of lists according according to the `RefitEvery` argument provided in the MultiGASRoll function. Each list is populated by three arguments: `vKappa` the intercept vector, `mA` the A system matrix, `mB` the B system matrix. By default, `do.list = FALSE`.

Author(s)

Leopoldo Catania

mGASSim *Class for Multivariate GAS Simulation*

Description

Class for multivariate GAS model simulation.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`ModelInfo`: Object of class `list`. Contains information about the multivariate GAS specification:

- iT: `numeric` Time length of simulated observations.
- iN: `numeric` Cross sectional dimension.
- iK : `numeric` number of (possibly) time–varying parameters implied by the distributional assumption.
- vKappa `numeric` vector of unconditional level for the reparametrized vector of parameters.
- mA `matrix` of coefficients of dimension iK x iK that premultiply the conditional score in the GAS updating recursion.
- mB `matrix` of autoregressive coefficients of dimension iK x iK.
- Dist `character` label of the conditional distribution, see DistInfo
- ScalingType `character` representing the scaling mechanism for the conditional score, see DistInfo

`GASDyn`: Object of class `list`. Contains: the series of simulated parameters (`GASDyn$mTheta`), the series of scaled scores (`GASDyn$mInnovation`), the series of unrestricted simulated parameters (`GASDyn$mTheta_tilde`), the series of log densities (`GASDyn$vLLK`), the log likelihood evaluated at its optimum value (`GASDyn$dLLK`)

`Data`: Object of class `matrix`. Matrix of dimension iN x iT of simulated data

Methods

- `show signature(object = 'mGASSim')`: Show summary.
- `plot signature(x = 'mGASSim',y = 'missing')`: Plot simulated data and parameters.
- `getFilteredParameters signature(object = 'mGASSim')`: Extract simulated parameters.
- `getObs signature(object = 'mGASSim')`: Extract simulated observations
- `coef signature(object = 'mGASSim')`: Extract delivered coefficients
- `getMoments signature(object = 'uGASFor')`: Extract simulated moments.

Author(s)

Leopoldo Catania

mGASSpec *Class for the Multivariate GAS model specification*

Description

Class for the Multivariate GAS model specification.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Spec`: Object of class `list`. Contains information abount the multivariate GAS specification:

- `Dist`: `character` Containing the conditional distribution assumption.
- `ScalingType`: `character` indicating the scaling mechanism for the conditional score.
- `GASPar`: `list` with elements: location, scale, correlation, shape.
- `ScalarParameters`: `logical` indicates if the parameters of the locations, scales and correlations dynamic have to be scalars or a diagonal matrices.

Methods

- `show` `signature(object = 'mGASSpec')`: Show summary.

Author(s)

Leopoldo Catania

MultiGASFit *Estimate multivariate GAS models*

Description

Estimate multivariate GAS models by Maximum Likelihood.

Usage

```
MultiGASFit(GASSpec, data, fn.optimizer = fn.optim, Compute.SE = TRUE,
          SE.A.tol = 1e-5, Compute.Hessian = FALSE)
```

Arguments

GASSpec	An object of the class mGASSpec created using the function MultiGASSpec
data	matrix (or something coercible to that using as.matrix()) of dimension TxN containing the multivariate time series of observations. It can also be an object of the class ts, xts or zoo.
fn.optimizer	function. This is a generic optimization function that can be provided by the user. By default fn.optimizer = fn.optim where fn.optim is a wrapper to the optim function. See Details for user defined optimization routines.
Compute.SE	logical. Should asymptotic Standard Errors be computed? By default Compute.SE = TRUE
Compute.Hessian	Logical, default FALSE. With the default optimizer, skip both the numerical Hessian used for curvature checks and the final returned Hessian. Set to TRUE to enable both. This option is independent of Compute.SE: OPG standard errors remain available without a Hessian. It applies to static initialization and the dynamic fit.
SE.A.tol	Finite nonnegative threshold for near-zero score loadings on their reported coefficient scale. If the full OPG fails its numerical checks, loadings with absolute value at most this threshold and their matching persistence coefficients are held fixed for conditional inference. The default is 1e-5; zero disables this fallback.

Details

For a fully static multivariate Gaussian model, the analytical sample-mean and sample-covariance MLE is used, with covariance divisor T. A full-rank sample covariance is required; the supplied optimizer is not called for this special case.

Maximum Likelihood estimation of GAS models is an on-going research topic. General results are reported by Blasques et al. (2014b), Blasques et al. (2014a) and Harvey (2013), while results for specific models have been derived by Blasques et al. (2014c) and Andres (2014).

Starting values for the optimizer are chosen in the following way: (i) estimate the static version of the model (i.e., with A = 0 and B = 0) and set the initial value of the intercept parameter accordingly, and (ii) perform a grid search for the coefficients contained in A and B. Further technical details are presented in Section 3.2 of Ardia et. al. (2016a).

The user is free to employ his/her own optimization routine via the fn.optimizer argument. fn.optimizer accepts a function object. The user provided optimizer has to satisfy strict requirements. The arguments of the fn.optimizer are : i) par0 a vector of starting values, ii) data the data provided, iii) GASSpec an object of the class mGASSpec, and iv) FUN the likelihood function. The output of fn.optimizer has to be a named list containing pars, the estimated numeric parameter vector; value, the summed negative log likelihood; and convergence, a numeric solver status with zero indicating success. An optional hessian element is retained as optimizer output but is not used for inference.

The Compute.Hessian option is forwarded to custom optimizers only when they explicitly declare an argument with that name. Legacy optimizers with the original four-argument interface remain supported; their internal Hessian calculations are controlled by the custom optimizer itself. The

`fn.solnp` wrapper only controls whether SOLNP's internally maintained Hessian approximation is returned. Its internal optimization updates are unaffected. For a static multivariate Gaussian fit, `Compute.Hessian = TRUE` computes a numerical Hessian at the analytical MLE.

When `Compute.SE = TRUE`, `numDeriv::jacobian` differentiates the vector of per-observation log likelihood contributions at the final estimate. The complete filter, including initialization, is rerun at each perturbed coefficient vector. If S is the resulting $T \times p$ score matrix, the outer-product information matrix is $S'S$ and the covariance estimate on the unrestricted optimizer scale is $(S'S)^{-1}$. There is no additional sample-size divisor. The delta method transforms this covariance to the reported coefficient scale. This is the OPG covariance for correctly specified maximum likelihood, not a sandwich covariance under misspecification.

If the full OPG is numerically singular or otherwise fails its checks, `SE.A.tol` permits conditional inference after excluding near-zero a_i coefficients and their matching b_i coefficients. The principal OPG submatrix for the remaining coefficients is inverted only if it passes the same checks. Standard errors, test statistics and p-values for the excluded coefficients are `NA`. Their estimates are unchanged; no refitting occurs. The remaining standard errors condition on the excluded coefficients at their fitted values and do not represent the full joint covariance or uncertainty from selecting the reduced set. A warning and a note in the fit display identify conditional inference. If the reduced matrix also fails, all inferential entries remain `NA`.

A missing convergence flag is recorded as unknown. A zero solver flag that fails the numerical gradient check is replaced by code 52. The original flag is retained in `Estimates$optimiser$solver_convergence`. The maximized objective is checked against the starting objective. Standard errors, test statistics and p-values are returned as `NA` when convergence checks fail or neither the full nor the permitted reduced OPG passes. No generalized inverse or automatic positive-definiteness repair is used. The coefficient estimates remain available. The score matrix, summed OPG matrix and its inverse are stored in `Estimates$optimiser$scores`, `Estimates$optimiser$opg` and `Estimates$optimiser$covariance`, respectively, when available. Under conditional inference, `covariance` is the reduced inverse embedded in a full-sized matrix with `NA` in excluded rows and columns. `inference_parameters` and `excluded_parameters` identify the retained and excluded coefficients. `opg` and `scores` still contain all coefficients. The `covariance_method` field is `"OPG"`; validation results are in `diagnostics`, including `full_information_ok`, `information_ok`, `conditional_inference`, `se_a_tol` and `inference_ok`. With `Compute.SE = FALSE`, these matrices are not computed and `inference_ok` is false.

By default, the `optim` optimizer with `method = "BFGS"` is employed.

Value

An object of the class mGASFit.

Author(s)

Leopoldo Catania

References

Ardia D, Boudt K and Catania L (2016a). "Generalized Autoregressive Score Models in R: The GAS Package." https://www.ssrn.com/abstract=2825380.

Blasques F, Koopman SJ, Lucas A (2014a). "Maximum Likelihood Estimation for Correctly Specified Generalized Autoregressive Score Models: Feedback Effects, Contraction Conditions and

Asymptotic Properties." techreport TI 14-074/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2332.

Blasques F, Koopman SJ, Lucas A (2014b). "Maximum Likelihood Estimation for Generalized Autoregressive Score Models." techreport TI 2014-029/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2286.

Blasques F, Koopman SJ, Lucas A, Schaumburg J (2014c). "Spillover Dynamics for Systemic Risk Measurement using Spatial Financial Time Series Models." techreport TI 2014-103/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2369.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Ghalanos A, Theussl S (2016). "Rsolnp: General Non-Linear Optimization using Augmented Lagrange Multiplier Method." https://cran.r-project.org/package=Rsolnp.

Gilbert P, Varadhan R (2016). numDeriv: Accurate Numerical Derivatives. R package 2016.8-1, https://CRAN.R-project.org/package=numDeriv.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Ye Y (1988). Interior Algorithms for Linear, Quadratic, and Linearly Constrained Convex Programming. Ph.D. thesis, Stanford University.

Examples

```
# Specify an GAS model with multivariate Student-t
# conditional distribution and time-varying scales and correlations

library("GAS")

data("StockIndices")

GASSpec = MultiGASSpec(Dist = "mvt", ScalingType = "Identity",
                       GASPar = list(scale = TRUE, correlation = TRUE))

Fit = MultiGASFit(GASSpec, StockIndices)

Fit
```

MultiGASFor *Forecast with multivariate GAS models*

Description

Forecast with multivariate GAS models. One-step ahead prediction of the conditional density is available in closed form. Multistep ahead prediction are performed by simulation as detailed in Blasques et al. (2016).

Usage

```
MultiGASFor(mGASFit, H = NULL, Roll = FALSE, out = NULL, B = 10000,
                     Bands = c(0.1, 0.15, 0.85, 0.9), ReturnDraws = FALSE)
```

Arguments

mGASFit	An object of the class mGASFit created using the function MultiGASFit
H	numeric Forecast horizon. Ignored if Roll = TRUE
Roll	logical Forecast should be made using a rolling procedure ? Note that if Roll = TRUE, then out has to be specified.
out	matrix of out of sample observation of dimension H x N for rolling forecast. N refers to the cross sectional dimension.
B	numeric Number of draws from the iH-step ahead distribution if Roll = FALSE.
Bands	numeric Vector of probabilities representing the confidence band levels for multistep ahead parameters forecasts. Only if Roll = FALSE.
ReturnDraws	logical Return the draws from the multistep ahead predictive distribution when Roll = FALSE ?

Value

An object of the class mGASFor

Author(s)

Leopoldo Catania

References

Blasques F, Koopman SJ, Lasak K, and Lucas, A (2016). "In-sample Confidence Bands and Out-of-Sample Forecast Bands for Time-Varying Parameters in Observation-Driven Models." International Journal of Forecasting, 32(3), 875-887. doi:10.1016/j.ijforecast.2016.04.002.

Examples

```
# Specify a GAS model with multivatiate  Student-t conditional
# distribution and time-varying scales and correlations.

# Stock returns forecast

set.seed(123)

data("StockIndices")
```

```
mY = StockIndices[, 1:2]

# Specification mvt
GASSpec = MultiGASSpec(Dist = "mvt", ScalingType = "Identity",
                       GASPar = list(location = FALSE, scale = TRUE,
                                     correlation = TRUE, shape = FALSE))

# Perform H-step ahead forecast with confidence bands

# Estimation
Fit = MultiGASFit(GASSpec, mY)

# Forecast
Forecast  = MultiGASFor(Fit, H = 50)

Forecast

# Perform 1-Step ahead rolling forecast

InSampleData  = mY[1:1000, ]
OutSampleData = mY[1001:2404, ]

# Estimation
Fit = MultiGASFit(GASSpec, InSampleData)

Forecast  = MultiGASFor(Fit, Roll = TRUE, out = OutSampleData)

Forecast
```

`MultiGASRoll` *Rolling forecast with multivariate GAS models*

Description

One-step ahead rolling forecasts with model re-estimation. The function also reports several quantity for backtesting for point and density forecasts.

Usage

```
MultiGASRoll(data, GASSpec, ForecastLength = 500, Nstart = NULL,
             RefitEvery = 23, RefitWindow = c("moving", "recursive"),
             cluster = NULL, Compute.SE = FALSE, ...)
```

Arguments

`data`	`matrix` of dimension (T + ForecastLength) x N containing the time series of observations.

GASSpec	An object of the class mGASSpec created using the function MultiGASSpec
ForecastLength	numeric Length of the out of sample
Nstart	numeric Number of observations in the first estimation sample. Ignored if ForecastLength is supplied.
RefitEvery	numeric Number of periods before model coefficients re-estimation.
RefitWindow	character Type of window. If RefitWindow = "recursive" all the observations are used when the model is re-estimated. If RefitWindow = "moving" old observations are eliminated.
cluster	A cluster object created calling using the paralell package. If supplied parallel processing is used to speed up the computations.
Compute.SE	logical. Should asymptotic Standard Errors be computed? By default Compute.SE = FALSE
...	Additional arguments for MultiGASFit

Details

Hessian computation is disabled by default. Pass Compute.Hessian = TRUE through ... to enable it for each refit, independently of Compute.SE.

If ForecastLength is explicitly supplied, it determines the forecast horizon and Nstart is ignored. Otherwise, a supplied Nstart is the number of observations in the first estimation sample, and the horizon is the sample size minus Nstart. If neither is supplied, ForecastLength defaults to 500. The resulting forecast horizon must be positive and smaller than the sample size. RefitEvery must be a positive integer.

Value

An object of the class mGASRoll

Author(s)

Leopoldo Catania

Examples

```
# Specify a GAS model with Multivariate Student-t conditional
# distribution and time-varying scale and correlation parameters

# stock returns Forecast

data("StockIndices")

mY = StockIndices[, 1:2]

# Specification mvt
GASSpec = MultiGASSpec(Dist = "mvt", ScalingType = "Identity",
                       GASPar = list(location = FALSE, scale = TRUE,
                                     correlation = TRUE, shape = FALSE))
```

```
# Perform 1-step ahead rolling forecast with refit
library(parallel)

Roll = MultiGASRoll(mY, GASSpec, ForecastLength = 250,
                        RefitEvery = 100, RefitWindow = c("moving"))

Roll
```

`MultiGASSim` *Simulate multivariate GAS processes*

Description

Simulate multivariate GAS processes.

Usage

```
MultiGASSim(fit = NULL, T.sim = 1000, N = NULL,
            kappa = NULL, A = NULL, B = NULL, Dist = NULL, ScalingType = NULL)
```

Arguments

`fit`	An estimated object of the class mGASFit. By default `fit = NULL`.
`T.sim`	`numeric` Length of the simulated time series.
`N`	An integer greater than or equal to 2 giving the cross-sectional dimension. Dimensions greater than four are supported.
`kappa`	`numeric` Vector of intercepts in the recursion for the reparametrised parameters. For a target state $\bar{f}$, use $\kappa = (I - B)\bar{f}$.
`A`	`matrix` of coefficients of dimension K x K that premultiply the conditional score in the GAS updating recursion, see Details.
`B`	`matrix` of autoregressive coefficients of dimension K x K, see Details.
`Dist`	`character` Label of the conditional distribution, see DistInfo.
`ScalingType`	`character` indicating the scaling mechanism for the conditional score. The choices are `"Identity"`, `"Inv"`, and `"InvSqrt"`; the last uses Cholesky whitening of the transformed score. See MultiGASSpec.

Details

The function permits to simulate from an estimated mGASFit object. If `fit` is not provided, the user can specify a GAS model via the additional arguments `kappa`, `A`, `B`, `Dist` and `ScalingType`.

All the information regarding the supported multivariate conditional distributions can be investigated using the DistInfo function. The model is specified as:

$$y_t \sim p(y|\theta_t)$$

where θ_t is the vector of parameters for the density $p(y|.)$. Note that, θ_t includes also those parameters that are not time-varying. The GAS recursion for θ_t is:

$$\theta_t = \Lambda(\tilde{\theta}_t)$$

$$\tilde{\theta}_t = \kappa + A * s_{t-1} + B * \tilde{\theta}_{t-1}$$

where $h(.)$ is the mapping function (see MultiMapParameters) and $\tilde{\theta}_t$ is the vector of reparametrised parameters. The process is initialized at $\tilde{\theta}_1 = (I - B)^{-1}\kappa$, where κ is the Kappa vector. The vector s_t is the scaled score of $p(y|.)$ with respect to $\tilde{\theta}_t$. See Ardia et. al. (2016a) for further details.

Value

An object of the class mGASSim

Author(s)

Leopoldo Catania

References

Ardia D, Boudt K and Catania L (2016a). "Generalized Autoregressive Score Models in R: The GAS Package." https://www.ssrn.com/abstract=2825380.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Examples

```
# Simulate from a GAS process with Multivariate Student-t conditional
# distribution, time-varying scales and correlations,
# and fixed location and shape parameters.
library("GAS")

set.seed(786)

T.sim = 1000  # Number of observations to simulate.
N     = 3     # Trivariate series.
Dist  = "mvt" # Conditional Multivariate Studen-t distribution.

# Specify the target initial distribution parameters.

Mu  = c(0.1, 0.2, 0.3) # Vector of location parameters (this is not transformed).
Phi = c(1.0, 1.2, 0.3) # Vector of scale parameters for the first, second and third variables.

# Distinct correlations, ordered as rho12, rho13, rho23.
Rho = c(0.1, 0.2, 0.3)
```

```
Theta = c(Mu, Phi, Rho, 7) # Vector of parameters such that the degrees of freedom are 7.

fbar = MultiUnmapParameters(Theta, Dist, N)

A = matrix(0, length(fbar), length(fbar))

# Update scales and correlations, do not update locations and shape parameters.
diag(A) = c(0, 0, 0, 0.05, 0.01, 0.09, 0.01, 0.04, 0.07, 0)

B = matrix(0, length(fbar), length(fbar))

# Update scales and correlations, do not update locations and shape parameters.
diag(B) = c(0, 0, 0, 0.7, 0.7, 0.5, 0.94, 0.97, 0.92, 0)

# Convert the target transformed state into the recursion intercept.
kappa = (diag(length(fbar)) - B) %*% fbar

Sim = MultiGASSim(fit = NULL, T.sim, N, kappa, A, B, Dist, ScalingType = "Identity")

Sim
```

`MultiGASSpec` *Multivariate GAS specification*

Description

Specify the conditional distribution, scaling mechanism and time-varying parameters for multivariate GAS models.

Usage

```
MultiGASSpec(Dist = "mvnorm", ScalingType = "Identity",
             GASPar = list(location = FALSE, scale = TRUE,
                           correlation = FALSE, shape = FALSE),
             ScalarParameters = TRUE)
```

Arguments

`Dist`	`character` indicating the label of the conditional distribution. Available distribution can be displayed using the function DistInfo. Default value `Dist = "mvnorm"`
.	
`ScalingType`	`character` indicating the scaling mechanism for the conditional score. The choices are `"Identity"`, `"Inv"`, and `"InvSqrt"`. The latter two use the full expected Fisher information in transformed state coordinates, including the cross terms with shape for Student-t models. For `"InvSqrt"`, Cholesky whitening is used, as in univariate models.

GASPar — list containing information about which parameters of the conditional distribution have to be time-varying. location = TRUE refers to the location parameters, scale = TRUE refers to the scale parameters, shape = TRUE refers to the shape parameter (e.g., the degree of freedom of the multivariate Student-t distribution), correlation = TRUE refers to the correlations. If the distribution specified in the Dist argument does not have, say, a shape parameter, the condition shape = TRUE is ignored.

ScalarParameters — logical indicating if the parameters of the locations, scales and correlations dynamic have to be scalars or a diagonal matrices. By default ScalarParameters = TRUE.

Details

All the information regarding the supported multivariate conditional distributions can be investigated using the DistInfo function.

Value

An object of the class mGASSpec

Author(s)

Leopoldo Catania

References

Creal D, Koopman SJ, Lucas A (2011). "A Dynamic Multivariate Heavy-Tailed Model for Time-Varying Volatilities and Correlations." Journal of Business & Economic Statistics, 29(4), 552-563. doi:10.1198/jbes.2011.10070.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Examples

```
# Specify a GAS model with multivariate Student-t
# conditional distribution and time-varying locations,
# scales and correlations  parameters but constant shape parameter.

library("GAS")

GASSpec = MultiGASSpec(Dist = "mvt", ScalingType = "Identity",
                       GASPar = list(location = TRUE, scale = TRUE,
                                     correlation = TRUE, shape = FALSE))

GASSpec
```

`MultiMapParameters` *Mapping function for multivariate distributions*

Description

Map unrestricted vector of parameters into the proper space. This function transforms the parameters updated using the GAS recursion into their proper space.

Usage

```
MultiMapParameters(Theta_tilde, Dist, N)
```

Arguments

`Theta_tilde`	`numeric` Vector of reparametrised parameters, see Details.
`Dist`	`character` Label of the conditional distribution, see DistInfo.
`N`	An integer greater than or equal to 2 giving the cross-sectional dimension. Dimensions greater than four are supported.

Details

The input vector contains, in order, N location parameters, N log-scale parameters, N(N-1)/2 correlation angles, and, for the multivariate Student-t distribution, one transformed degrees-of-freedom parameter.

Locations are unchanged and scales are obtained by exponentiation, subject to the package's numerical safeguards. The Student-t degrees-of-freedom parameter uses the bounded transformation implemented by the package. Use `MultiUnmapParameters` to construct the transformed vector from distribution parameters.

The correlation angles are ordered as $\phi_{12}, \phi_{13}, \ldots, \phi_{1N}, \phi_{23}, \ldots, \phi_{2N}, \ldots, \phi_{N-1,N}$. They define an upper-triangular matrix X through

$$X_{rb} = \cos(\phi_{rb}) \prod_{s<r} \sin(\phi_{sb}), \quad r < b,$$

$$X_{bb} = \prod_{s<b} \sin(\phi_{sb}), \qquad X_{rb} = 0, \quad r > b,$$

where empty products equal one. The correlation matrix is

$$R = X^\top X.$$

The returned correlation entries follow the same pair ordering. They contain only the distinct off-diagonal entries, rather than the full vectorization of R.

This construction gives a positive semidefinite matrix with unit diagonal. It is positive definite when all diagonal entries of X are nonzero. In particular, angles strictly between zero and pi give a

positive definite matrix in exact arithmetic. Angles near singular configurations can cause numerical ill-conditioning.

The correlation Jacobian is computed analytically for every supported dimension by differentiating $R = X^\top X$.

Value

A `numeric` vector of parameters.

Author(s)

Leopoldo Catania

Examples

```
N <- 5L
Dist <- "mvt"

# Equicorrelation matrix with correlation 0.2.
# Parameter order: locations, scales, correlations, degrees of freedom.
Theta <- c(
    rep(0, N),
    rep(1, N),
    rep(0.2, N * (N - 1L) / 2L),
    8
)

Theta_tilde <- MultiUnmapParameters(Theta, Dist, N)
Theta_back <- MultiMapParameters(Theta_tilde, Dist, N)

stopifnot(max(abs(Theta_back - Theta)) < 1e-10)
```

`MultiUnmapParameters` *Inverse of MultiMapParameters*

Description

Transform distribution parameters into the unrestricted parameters. The unrestricted vector of parameters is updated using the GAS recursion.

Usage

```
MultiUnmapParameters(Theta, Dist, N)
```

Arguments

`Theta`	`numeric` Vector parameters, see Details.
`Dist`	`character` Label of the conditional distribution, see DistInfo.
`N`	An integer greater than or equal to 2 giving the cross-sectional dimension. Dimensions greater than four are supported.

Details

The input vector contains, in order, N locations, N positive scales, N(N-1)/2 distinct correlations, and, for the multivariate Student-t distribution, one degrees-of-freedom parameter.

Correlations must be ordered as $\rho_{12}, \rho_{13}, \ldots, \rho_{1N}, \rho_{23}, \ldots, \rho_{2N}, \ldots, \rho_{N-1,N}$. Together with a unit diagonal, they must define a positive definite correlation matrix.

The function returns locations, log-scales, correlation angles, and, where applicable, transformed degrees of freedom. The degrees of freedom must lie strictly inside the bounds used by the package.

The inverse correlation mapping selects angles in the interval $(0, \pi)$. Because the forward mapping is periodic, transforming arbitrary angles to correlations and back need not recover the original angles. Transforming admissible distribution parameters to the unrestricted representation and back recovers the original distribution parameters up to numerical precision.

See `MultiMapParameters` for the hyperspherical mapping.

Value

A `numeric` vector of parameters.

Author(s)

Leopoldo Catania

Examples

```
# Unmap parameters for the Multivariate Student-t distribution with N=3
library(GAS)

N = 3

Dist = "mvt"

# Vector of location parameters (this is not transformed).
Mu  = c(0.1, 0.2, 0.3)

# Vector of scales parameters for the firs, second and third variables.
Phi = c(1.0, 1.2, 0.3)

# This represents the distinct correlations in the order \(\rho_{12},\rho_{13},\rho_{23}\).
# Note that is up to the user to ensure that vec(R) implies a proper correlation matrix
Rho = c(0.1, 0.2, 0.3)

# Vector of parameters such that the degrees of freedom are 7.
Theta = c(Mu, Phi, Rho, 7)

Theta_tilde = MultiUnmapParameters(Theta, Dist, N)

Theta_tilde

# It works
all(abs(MultiMapParameters(Theta_tilde, Dist, N) - Theta) < 1e-10)
```

NumericalBounds *Numerical bounds used in parameter transformations*

Description

Returns the numerical bounds used by the package's parameter transformations.

Usage

```
UpperNu(Dist = "std")
LowerNu(Dist = "std")
UpperA()
LowerA()
UpperB()
LowerB()
```

Arguments

Dist	Distribution label: "std", "sstd", "ast", "ast1", "ghskt", or "mvt".

Details

LowerNu and UpperNu are R wrappers around the same C++ functions used in the forward mappings, inverse mappings and mapping Jacobians. The default distribution is "std"; calls without arguments remain supported. Unsupported distribution labels cause an error.

The lower bound is 0.001 for std, ast, ast1, and mvt; 2 for sstd; and 4 for ghskt. Both degrees-of-freedom parameters of ast use the same bounds. The upper bound is 50 for all six distributions. The mapped parameter lies strictly between the bounds. The lower bound of 0.001 is a numerical floor. The higher bounds for sstd and ghskt reflect their current mean/variance-standardized parameterizations. Bounds are not increased merely to ensure that third or fourth moments exist.

UpperA and LowerA return bounds for the diagonal score-loading coefficients in the GAS recursion. UpperB and LowerB return bounds for the diagonal autoregressive coefficients.

Models fitted with earlier shape mappings should be refitted before using this implementation: the stored unrestricted shape states have changed meaning.

Value

A numeric scalar giving the requested bound.

Author(s)

Leopoldo Catania

Examples

```
LowerNu()           # std: 0.001
LowerNu("sstd")     # 2
LowerNu("ghskt")    # 4
UpperNu("mvt")      # 50
UpperA()
LowerA()
UpperB()
LowerB()
```

PIT_test *Goodness of fit for conditional densities*

Description

This function implements density goodness of fit procedure of Diebold et al. (1998).

Usage

```
PIT_test(U, G = 20, alpha = 0.05, plot = FALSE)
```

Arguments

U	numeric Vector of Probability Integral Transformation (PIT).
G	Integer of at least two giving the number of equal-width histogram bins on [0,1]. The supplied value is retained for all sample sizes.
alpha	Significance level strictly between zero and one.
plot	Logical; whether to display the PIT histogram. Default is FALSE.

Details

All input values must be finite and lie in [0,1]. The histogram uses G fixed equal-width bins over this entire interval. Its Pearson statistic uses expected counts length(U)/G and a chi-squared reference distribution with G-1 degrees of freedom. If an expected count is below five, the statistic is returned but its critical value and p-value are NA, with an explanation in reason. The plotted count limits are approximate two-sided marginal limits with binomial variance; they are not simultaneous limits across bins.

The dependence diagnostics regress each of the first four centered PIT powers on an intercept and its twenty lags. They require more than 41 observations and a full-rank design. Unavailable tests are reported as NA, with a reason, and do not abort model fitting.

For discrete distributions, ordinary CDF values F(Y) are not uniform. The package computes randomized PITs for ber, poi, negbin and skellam:

$$U_t = F_t(Y_t - 1) + V_t\{F_t(Y_t) - F_t(Y_t - 1)\},$$

where the V variables are independent Uniform(0,1) random variables. Set `set.seed()` before fitting or forecasting for reproducibility. This consumes random numbers. Calls to `pit()` extract the stored realization without drawing new random variables.

The reference distributions are asymptotic diagnostics. Estimated in-sample parameters and dependence can affect their finite-sample calibration; randomized PITs correct the discrete-CDF issue but do not remove parameter-estimation effects.

This function implements density goodness of fit procedure of Diebold et al. (1998). The test relays on the result that, if the series of estimated conditional distributions is the true one, then the PIT series evaluated accordingly are iid Unif(0, 1) distributed. The test of the iid Uniform(0, 1) assumption consists of two parts. The first part concerns the independent assumption, and it tests if all the conditional moments of the data, up to the fourth one, have been accounted for by the model, while the second part checks if the conditional distribution assumption is reliable by testing if the PITs are Uniform over the interval (0, 1). See also Jondeau and Rockinger (2006) and Vlaar and Palm (1993).

Value

A `list` with elements: (i) `Hist` and (ii) `IID`. The fist element `Hist` concerns the test of the unconditional assumption of uniformity of the PIT, it is a `list` with elements:

- `test` Statistic test.
- `crit` The critical value of the test.
- `pvalue` The pvalue of the test.
- `hist` The histogram, evaluated using the hist function.
- `confidence` Approximated asymptotic confidence level.

The second element `IID` concerns the iid assumption, it is a `list` with elements:

- `test` A named `numeric` vector with elements: `test1`, `test2`, `test3`, `test4` representing the Lagrange Multiplier test for the first four conditional moments of the PITs.
- `crit` The critical value of the test.
- `pvalue` A named `numeric` vector with elements: `pvalue1`, `pvalue2`, `pvalue3`, `pvalue4` representing the pvalues of the Lagrange Multiplier test for the first four conditional moments of the PITs.

Author(s)

Leopoldo Catania

References

Diebold FX, Gunther TA and Tay AS (1998). "Evaluating Density Forecasts with Applications to Financial Risk Management." International Economic Review, 39(4), 863-883.

Jondeau E and Rockinger M (2006). "The Copula-Garch Model of Conditional Dependencies: An International Stock Market Application." Journal of International Money and Finance, 25(5), 827-853. doi:10.1016/j.jimonfin.2006.04.007.

Vlaar PJ and Palm FC (1993). "The Message in Weekly Exchange Rates in the European Monetary System: Mean Reversion, Conditional Heteroscedasticity, and Jumps." Journal of Business & Economic Statistics, 11(3), 351-360. doi:10.1080/07350015.1993.10509963.

Examples

```
data("StockIndices")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = FALSE, scale = TRUE,
                                   shape = FALSE))

FTSEMIB = StockIndices[, "FTSEMIB"]

Fit = UniGASFit(GASSpec, FTSEMIB)

U = pit(Fit)

Test = PIT_test(U, G = 20, alpha = 0.05, plot = TRUE)
```

`plot-methods` *Plot output from an object of the GAS package.*

Description

This method provides plot functionalities for the uGASFit, mGASFit, uGASSim, mGASSim, uGASFor, mGASFor, uGASRoll and mGASRoll objects defined in the GAS package.

Usage

```
  ## S4 method for signature 'uGASFit,missing'
plot(x, which = NULL)
  ## S4 method for signature 'mGASFit,missing'
plot(x, which = NULL)
  ## S4 method for signature 'uGASSim,missing'
plot(x, which = NULL)
  ## S4 method for signature 'mGASSim,missing'
plot(x, which = NULL)
  ## S4 method for signature 'uGASFor,missing'
plot(x, which = NULL)
  ## S4 method for signature 'mGASFor,missing'
plot(x, which = NULL)
  ## S4 method for signature 'uGASRoll,missing'
plot(x, which = NULL)
  ## S4 method for signature 'mGASRoll,missing'
plot(x, which = NULL)
  PlotMenu(x)
```

Arguments

x	objects of class uGASFit, mGASFit, uGASSim, mGASSim, uGASFor, mGASFor, uGASRoll, mGASRoll.
which	Numeric menu option to plot. If NULL, display the interactive menu. See Details.

Details

plot accepts the additional numeric argument which. By default which = NULL. If which is provided, plot() does not show the interactive menu and plot the corresponding option. The available options for each object class is printed by the function PlotMenu(x). By default which = NULL, that is, plot() display an interactive menu.

Value

Displays a plot of an object of class uGASFit, mGASFit, uGASSim, mGASSim, uGASFor, mGASFor, uGASRoll, mGASRoll.

Author(s)

Leopoldo Catania

Examples

```
## Plot filtered estimates of a GAS model estimated on the
## Quarterly logarithmic change in percentage points of the Consumer Price Index data set (cpichg)
library("GAS")

data("cpichg")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE, scale = TRUE,
                                   shape = FALSE))

Fit = UniGASFit(GASSpec, cpichg)

plot(Fit, which = 1)
```

---

sp500ret — *Data: Daily logarithmic returns in percentage points of the S&P500 index from 1950-01-04 to 2016-06-24*

Description

Daily logarithmic returns in percentage points of the S&P500 index from 1950-01-04 to 2016-06-24 obtained from yahoo finance.

Usage

```
data("sp500ret")
```

Format

A xts object of dimension 16,727 x 1 containing the daily logarithmic returns in percentage points from 1950-01-04 to 2016-06-24.

Source

Yahoo Finance

---

sp500rv *Data: SP500 Daily 5-minute Realized Volatility*

Description

Oxford-Man Institute daily 5-minute realized volatility for the SP500 Index, covering January 2000 to June 2016.

Usage

```
data("sp500rv")
```

Format

An xts object containing 4,310 observations.

StockIndices *Data: Daily logarithmic returns in percentage points of the DAX, FTSEMIB and CAC40 from 2007-01-03 to 2016-06-24*

Description

Daily logarithmic returns in percentage points of the DAX, FTSEMIB and CAC40 from 2007-01-03 to 2016-06-24 obtained from Yahoo.

Usage

```
data("StockIndices")
```

Format

A `matrix` object of dimension 2,445 x 3 containing the daily logarithmic returns in percentage points from 2007-01-03 to 2016-06-24. Missing values are simply removed.

References

Yahoo finance.

`tqdata` *Data from Bien et al (2011).*

Description

From the readme.bnp.txt file in the JAE Data Archive available at http://qed.econ.queensu.ca/jae/2011-v26.4/bien-nolte-pohlmeier/:

The high-frequency data used in the paper come from the Trades and Quotation (TAQ) database. The data contains time-stamped quotations of Citicorp stock traded at the NYSE over the period from 20th February to 23rd February 2001.

In the study, 30-second bid and ask quote changes are constructed from the irregularly-spaced quote data. The study covers observations recorded from 9:35 EST until 16:00 EST.

The data contains 3080 rows and eight columns - in order:

1. year 2. month 3. day 4. time in number of seconds after the 9:35 EST 5. best ask quote 6. best bid quote 7. 30-second change of the ask quote in number of ticks 8. 30-second change of the bid quote in number of ticks.

Usage

```
data("tqdata")
```

Format

A data.frame object containing 3,080 observat. ions.

References

Bien K, Nolte, I, Pohlmeier W (2011). "An Inflated Multivariate Integer Count Hurdle Model: An Application to Bid and Ask Quote Dynamics". Journal of Applied Econometrics, 26(4), 669-707. doi:10.1002/jae.1122

uGASFit *Class for the univariate GAS fitted object*

Description

Class for the univariate GAS fitted object.

Details

The show and summary methods report *Average Filtered Parameters*: the arithmetic average of each mapped distribution parameter in GASDyn$mTheta over the first ModelInfo$iT columns. The final column, containing the next-period forecast, is excluded.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

ModelInfo: Object of class list. Contains information about the GAS specification:

- Spec: object of the class uGASSpec containing the GAS specification.
- iT: numeric number of observation.
- elapsedTime: numeric elapsed Time in seconds.

GASDyn: Object of class list. Contains: the series of filtered dynamic (GASDyn$mTheta) for the time–varying parameters, the series of scaled scores (GASDyn$mInnovation), the series of unrestricted filtered parameters (GASDyn$mTheta_tilde), the series of log densities (GASDyn$vLLK), the log likelihood evaluated at its optimum value (GASDyn$dLLK).

Estimates: Object of class list. Contains: lParList list of estimated parameters, optimiser object delivered from the optimization function, StaticFit ML estimates for the constant model, Inference inferential results for the estimated parameters.

Data: The user's data.

Testing: Statistical tests results.

Methods

- show signature(object = 'uGASFit'): print object information.
- summary signature(object = 'uGASFit'): Show summary.
- plot signature(x = 'uGASFit',y = 'missing'): Plot filtered dynamic and other estimated quantities.
- getFilteredParameters signature(object = 'uGASFit'): Extract filtered parameters.
- getObs signature(object = 'uGASFit'): Extract original observations.
- coef signature(object = 'uGASFit'): Returns a named vector of estimated coefficients. Also accepts the additional logical argument do.list. If do.list = TRUE, estimated coefficients are organized in a list with arguments: vKappa the intercept vector, mA the A system matrix, mB the B system matrix. By default, do.list = FALSE.

- pit signature(object = 'uGASFit'): Extract Probability Integral Transformation.
- getMoments signature(object = 'uGASFit'): Extract conditional moments.
- residuals signature(object = 'uGASFit'): Extract the residuals. Also accepts the additional logical argument standardize. If standardize = TRUE, residuals are standardized by the filtered standard deviation. By default standardize = FALSE.
- convergence signature(object = 'uGASFit'): Extract convergence information.
- quantile signature(object = 'uGASSim'): Compute quantiles of the filtered estimated density at each point in time. It accepts the additional argument probs representing the vector of probabilities.
- ES signature(object = 'uGASSim'): Compute Expected Shortfall of the filtered estimated density at each point in time. It accepts the additional argument probs representing the vector of probabilities.

Note

For Bernoulli, Poisson, negative-binomial and Skellam models, stored PITs are randomized. Set set.seed() before constructing the object for reproducibility. See PIT_test.

Author(s)

Leopoldo Catania

---

uGASFor *Class for the univariate GAS forecast object*

Description

Class for the univariate GAS forecast object.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

Forecast: Object of class list. Contains forecasts:

- PointForecast: matrix with parameters forecasts.
- Moments: matrix with centered moments forecasts.
- vLS: numeric Log Score (Predictive Log Likelihood).
- vU: numeric Out-of-sample Probability Integral Transformation (PIT).

Bands: array with confidence bands parameters forecasts. Available only if Roll = TRUE.

Draws: If ReturnDraws = TRUE it is a iH x iB matrix of draws from the predictive distribution.

Info: list with forecast information.

Data: list with original data.

Methods

- show `signature(object = 'uGASFor')`: Show summary.
- plot `signature(x = 'uGASFor', y = 'missing')`: Plot forecasted quantities.
- getForecast `signature(object = 'uGASFor')`: Extract parameters forecast.
- getObs `signature(object = 'uGASFor')`: Extract original observations.
- pit `signature(object = 'uGASFor')`: Extract Probability Integral Transformation, only if `Roll = TRUE`.
- quantile `signature(object = 'uGASFor')`: Extract quantile forecasts. For multistep ahead prediction quantiles are computed by simulation and `ReturnDraws = TRUE` should have been selected. It accepts the additional argument `probs` representing the vector of probabilities.
- ES `signature(object = 'uGASFor')`: Extract Expected Shortfall forecasts. For multistep ahead prediction quantiles are computed by simulation and `ReturnDraws = TRUE` should have been selected. It accepts the additional argument `probs` representing the vector of probabilities.
- getMoments `signature(object = 'uGASFor')`: Extract moments forecasts.
- LogScore `signature(object = 'uGASFor')`: Extract Log Scores.

Note

For Bernoulli, Poisson, negative-binomial and Skellam models, stored PITs are randomized. Set `set.seed()` before constructing the object for reproducibility. See `PIT_test`.

Author(s)

Leopoldo Catania

---

uGASRoll *Class for the univariate GAS rolling object*

Description

Class for the univariate GAS rolling object.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Forecast`**:** Object of class `list`. Contains forecasts:

- `PointForecast`: `matrix` with parameters forecasts.
- `Moments`: `matrix` with centered moments forecasts.
- `vLS`: `numeric` Log Score (Predictive Log Likelihood).
- `vU`: `numeric` Out-of-sample Probability Integral Transformation (PIT).

`Info`: `list` with forecast information.

`Data`: `list` with original data.

`Testing`: Statistical tests results.

Methods

- `show signature(object = 'uGASRoll')`: Show summary.
- `plot signature(x = 'uGASRoll', y = 'missing')`: Plot forecasted quantities.
- `getForecast signature(object = 'uGASRoll')`: Extract parameters forecast.
- `getObs signature(object = 'uGASRoll')`: Extract original observations.
- `pit signature(object = 'uGASRoll')`: Extract Probability Integral Transformation, only if `Roll = TRUE`
- `quantile signature(object = 'uGASRoll')`: Extract quantile forecasts. It accepts the addional argument `probs` representing the vector of probabilities.
- `ES signature(object = 'uGASRoll')`: Extract Expected Shortfall forecasts. It accepts the addional argument `probs` representing the vector of probabilities.
- `getMoments signature(object = 'uGASRoll')`: Extract moments forecasts.
- `LogScore signature(object = 'uGASRoll')`: Extract Log Scores.
- `residuals signature(object = 'uGASRoll')`: Extract the forecast errors. Also accepts the additional logical argument `standardize`. If `standardize = TRUE`, forecast errors are standardized by the forecast standard deviation. By default `standardize = FALSE`.
- `coef signature(object = 'uGASFit')`: Returns a matrix of estimated coefficients. Each row of the matrix corresponds to a refit of the model during the forecast period according to the `RefitEvery` argument provided in the UniGASRoll function. Also accepts the additional logical argument `do.list`. If `do.list = TRUE`, estimated coefficients are organized in a list of lists according according to the `RefitEvery` argument provided in the UniGASRoll function. Each list is populated by three arguments: `vKappa` the intercept vector, `mA` the A system matrix, `mB` the B system matrix. By default, `do.list = FALSE`.

Note

For Bernoulli, Poisson, negative-binomial and Skellam models, stored PITs are randomized. Set `set.seed()` before constructing the object for reproducibility. See `PIT_test`.

Author(s)

Leopoldo Catania

uGASSim *Class for Univariate GAS Simulation*

Description

Class for Univariate GAS model Simulation.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`ModelInfo`**:** Object of class `list`. Contains information about the univariate GAS specification:

- `iT` `numeric` Time length of simulated observations.
- `iK` `numeric` Number of (possibly) time-varying parameters implied by the distributional assumption.
- `vKappa` `numeric` Vector of unconditional level for the reparametrised vector of parameters.
- `mA` `matrix` Of coefficients of dimension iK x iK that premultiply the conditional score in the GAS updating recursion.
- `mB` `matrix` Of autoregressive coefficients of dimension iK x iK.
- `Dist` `character` Label of the conditional distribution, see DistInfo
- `ScalingType` `character` Representing the scaling mechanism for the conditional score, see DistInfo.

`GASDyn`**:** Object of class `list`. Contains: the series of simulated parameters (`GASDyn$mTheta`), the series of scaled scores (`GASDyn$mInnovation`), the series of unrestricted simulated parameters (`GASDyn$mTheta_tilde`), the series of log densities (`GASDyn$vLLK`), the log likelihood evaluated at its optimum value (`GASDyn$dLLK`).

`Data`**:** Object of class `numeric`. Vector of length `iT` of simulated data.

Methods

- `show` `signature(object = 'uGASSim')`: Show summary.
- `plot` `signature(x = 'uGASSim', y = 'missing')`: Plot simulated data and parameters.
- `getFilteredParameters` `signature(object = 'uGASSim')`: Extract simulated parameters.
- `getObs` `signature(object = 'uGASSim')`: Extract simulated observations.
- `coef` `signature(object = 'uGASSim')`: Extract delivered coefficients.
- `quantile` `signature(object = 'uGASSim')`: Compute quantiles of the filtered simulated density at each point in time. It accepts the additional argument `probs` representing the vector of probabilities.
- `ES` `signature(object = 'uGASSim')`: Compute the Expected Shortfall of the filtered simulated density at each point in time. It accepts the additional argument `probs` representing the vector of probabilities.

Author(s)

Leopoldo Catania

---

uGASSpec *Class for the univariate GAS model specification*

Description

Class for the univariate GAS model specification.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`Spec`**:** Object of class `list`. Contains information about the univariate GAS specification:

- `Dist`: `character` containing the conditional distribution assumption.
- `ScalingType`: `character` indicating the scaling mechanism for the conditional score.
- `iK`: `numeric` representing the number of (possibly) time-varying parameters implied by the distributional assumption.
- `GASPar` `list` with elements: `location`, `scale`, `skewness`, `shape`, `shape2`.

Methods

- `show` `signature(object = 'uGASSpec')`: Show summary.

Author(s)

Leopoldo Catania

UniGASFit *Estimate univariate GAS models*

Description

Estimate univariate GAS models by Maximum Likelihood.

Usage

```
UniGASFit(GASSpec, data, fn.optimizer = fn.optim, Compute.SE = TRUE,
          SE.A.tol = 1e-5, Compute.Hessian = FALSE)
```

Arguments

GASSpec	An object of the class uGASSpec created using the function UniGASSpec
.	
data	numeric vector of length Tx1 containing the time series of observations. It can also be an object of the class ts, xts or zoo.
fn.optimizer	function. This is a generic optimization function that can be provided by the user. By default fn.optimizer = fn.optim where fn.optim is a wrapper to the optim function. See Details for user defined optimization routines.
Compute.SE	logical. Should asymptotic Standard Errors be computed? By default Compute.SE = TRUE
Compute.Hessian	Logical, default FALSE. With the default optimizer, skip both the numerical Hessian used for curvature checks and the final returned Hessian. Set to TRUE to enable both. This option is independent of Compute.SE: OPG standard errors remain available without a Hessian. It applies to static initialization and the dynamic fit.
SE.A.tol	Finite nonnegative threshold for near-zero score loadings on their reported coefficient scale. If the full OPG fails its numerical checks, loadings with absolute value at most this threshold and their matching persistence coefficients are held fixed for conditional inference. The default is 1e-5; zero disables this fallback.

Details

Maximum Likelihood estimation of GAS models is an on-going research topic. General results are reported by Blasques et al. (2014b), Blasques et al. (2014a) and Harvey (2013), while results for specific models have been derived by Blasques et al. (2014c) and Andres (2014).

Starting values for the optimizer are chosen in the following way: (i) estimate the static version of the model (i.e., with A = 0 and B = 0) and set the initial value of the intercept parameter accordingly, and (ii) perform a grid search for the coefficients contained in A and B. Further technical details are presented in Section 3.2 of Ardia et. al. (2016a).

The user is free to employ his/her own optimization routine via the fn.optimizer argument. fn.optimizer accepts a function object. The user provided optimizer has to satisfy strict requirements. The arguments of the fn.optimizer are : i) par0 a vector of starting values, ii) data the data provided, iii) GASSpec an object of the class uGASSpec, and iv) FUN the likelihood function. The output of fn.optimizer has to be a named list containing pars, the estimated numeric parameter vector; value, the summed negative log likelihood; and convergence, a numeric solver status with zero indicating success. An optional hessian element is retained as optimizer output but is not used for inference.

The Compute.Hessian option is forwarded to custom optimizers only when they explicitly declare an argument with that name. Legacy optimizers with the original four-argument interface remain supported; their internal Hessian calculations are controlled by the custom optimizer itself. The fn.solnp wrapper only controls whether SOLNP's internally maintained Hessian approximation is returned. Its internal optimization updates are unaffected. For a static multivariate Gaussian fit, Compute.Hessian = TRUE computes a numerical Hessian at the analytical MLE.

When `Compute.SE = TRUE`, `numDeriv::jacobian` differentiates the vector of per-observation log likelihood contributions at the final estimate. The complete filter, including initialization, is rerun at each perturbed coefficient vector. If S is the resulting $T \times p$ score matrix, the outer-product information matrix is $S'S$ and the covariance estimate on the unrestricted optimizer scale is $(S'S)^{-1}$. There is no additional sample-size divisor. The delta method transforms this covariance to the reported coefficient scale. This is the OPG covariance for correctly specified maximum likelihood, not a sandwich covariance under misspecification.

If the full OPG is numerically singular or otherwise fails its checks, `SE.A.tol` permits conditional inference after excluding near-zero a_i coefficients and their matching b_i coefficients. The principal OPG submatrix for the remaining coefficients is inverted only if it passes the same checks. Standard errors, test statistics and p-values for the excluded coefficients are `NA`. Their estimates are unchanged; no refitting occurs. The remaining standard errors condition on the excluded coefficients at their fitted values and do not represent the full joint covariance or uncertainty from selecting the reduced set. A warning and a note in the fit display identify conditional inference. If the reduced matrix also fails, all inferential entries remain `NA`.

A missing convergence flag is recorded as unknown. A zero solver flag that fails the numerical gradient check is replaced by code 52. The original flag is retained in `Estimates$optimiser$solver_convergence`. The maximized objective is checked against the starting objective. Standard errors, test statistics and p-values are returned as `NA` when convergence checks fail or neither the full nor the permitted reduced OPG passes. No generalized inverse or automatic positive-definiteness repair is used. The coefficient estimates remain available. The score matrix, summed OPG matrix and its inverse are stored in `Estimates$optimiser$scores`, `Estimates$optimiser$opg` and `Estimates$optimiser$covariance`, respectively, when available. Under conditional inference, `covariance` is the reduced inverse embedded in a full-sized matrix with `NA` in excluded rows and columns. `inference_parameters` and `excluded_parameters` identify the retained and excluded coefficients. `opg` and `scores` still contain all coefficients. The `covariance_method` field is `"OPG"`; validation results are in `diagnostics`, including `full_information_ok`, `information_ok`, `conditional_inference`, `se_a_tol` and `inference_ok`. With `Compute.SE = FALSE`, these matrices are not computed and `inference_ok` is false.

By default, the `optim` optimizer with `method = "BFGS"` is employed.

Value

An object of the class uGASFit

Author(s)

Leopoldo Catania

References

Ardia D, Boudt K and Catania L (2016a). "Generalized Autoregressive Score Models in R: The GAS Package." https://www.ssrn.com/abstract=2825380.

Blasques F, Koopman SJ, Lucas A (2014a). "Maximum Likelihood Estimation for Correctly Specified Generalized Autoregressive Score Models: Feedback Effects, Contraction Conditions and Asymptotic Properties." techreport TI 14-074/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2332.

Blasques F, Koopman SJ, Lucas A (2014b). "Maximum Likelihood Estimation for Generalized Autoregressive Score Models." techreport TI 2014-029/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2286.

Blasques F, Koopman SJ, Lucas A, Schaumburg J (2014c). "Spillover Dynamics for Systemic Risk Measurement using Spatial Financial Time Series Models." techreport TI 2014-103/III, Tinbergen Institute. https://tinbergen.nl/discussionpaper/?paper=2369.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Gilbert P, Varadhan R (2016). numDeriv: Accurate Numerical Derivatives. R package 2016.8-1, https://CRAN.R-project.org/package=numDeriv.

Ghalanos A, Theussl S (2016). "Rsolnp: General Non-Linear Optimization using Augmented Lagrange Multiplier Method." https://cran.r-project.org/package=Rsolnp.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Ye Y (1988). Interior Algorithms for Linear, Quadratic, and Linearly Constrained Convex Programming. Ph.D. thesis, Stanford University.

Examples

```
library("GAS")

# Use a short sample to keep the examples quick to run.
data("sp500ret")
vY <- tail(sp500ret, 500)
GASSpec <- UniGASSpec(
  Dist = "std", ScalingType = "Identity",
  GASPar = list(location = FALSE, scale = TRUE, shape = FALSE)
)
Fit <- UniGASFit(GASSpec, vY)
Fit

# Custom optimizer: Nelder-Mead followed by gradient-based refinement.
# Minimize the average negative log likelihood for numerical scaling,
# but return the summed negative log likelihood required by UniGASFit.
fn.NM.optim <- function(par0, data, GASSpec, FUN) {
  objective <- function(par) FUN(par, data = data, GASSpec = GASSpec)
  nm <- stats::optim(
    par0, function(par) objective(par) / length(data),
    method = "Nelder-Mead",
    control = list(maxit = 2000, reltol = 1e-10), hessian = FALSE
  )
  start <- par0
  if (is.finite(nm$value) && objective(nm$par) < objective(par0)) {
    start <- stats::setNames(nm$par, names(par0))
```

```
  }
  fn.optim(start, data, GASSpec, FUN, Compute.Hessian = FALSE)
}
Fit.NM.optim <- UniGASFit(GASSpec, vY, fn.optimizer = fn.NM.optim)
Fit.NM.optim

# Custom optimizer: random restarts with gosolnp(), followed by
# gradient-based refinement. Retain GAS's starting values as a candidate.

fn.gosolnp <- function(par0, data, GASSpec, FUN) {
  # Static initialization uses the default optimizer.
  if (is(GASSpec, "list")) {
    return(fn.optim(par0, data, GASSpec, FUN, Compute.Hessian = FALSE))
  }
  objective <- function(par) FUN(par, data = data, GASSpec = GASSpec)
  global <- Rsolnp::gosolnp(
    pars = par0, fun = FUN, data = data, GASSpec = GASSpec,
    n.sim = 100, n.restarts = 2, rseed = 123,
    LB = c(-5, -2, -2, -2), UB = c(5, 8, 3, 5),
    control = list(trace = 0)
  )
  start <- par0
  if (length(global$pars) == length(par0) &&
      all(is.finite(global$pars)) &&
      is.finite(objective(global$pars)) &&
      objective(global$pars) < objective(par0)) {
    start <- stats::setNames(global$pars, names(par0))
  }
  fn.optim(start, data, GASSpec, FUN, Compute.Hessian = FALSE)
}

data("Goals", package = "GAS")
GASSpec <- UniGASSpec(
  Dist = "negbin", ScalingType = "Inv",
  GASPar = list(location = TRUE, scale = FALSE)
)
vY <- as.numeric(stats::na.omit(Goals[, 1]))
UniGASFit(GASSpec, vY, fn.optimizer = fn.gosolnp)
```

UniGASFor *Forecast with univariate GAS models*

Description

Forecast with univariate GAS models. The one-step ahead prediction of the conditional density is available in closed form. The multi-step ahead prediction is performed by simulation as detailed in Blasques et al. (2016).

Usage

```
UniGASFor(uGASFit, H = NULL, Roll = FALSE, out = NULL, B = 10000,
                    Bands = c(0.1, 0.15, 0.85, 0.9), ReturnDraws = FALSE)
```

Arguments

uGASFit	An object of the class uGASFit created using the function UniGASFit.
H	numeric Forecast horizon. Ignored if Roll = TRUE.
Roll	logical Forecast should be made using a rolling procedure ? Note that, if Roll = TRUE, then out has to be specified.
out	numeric Vector of out-of-sample observation for rolling forecast.
B	numeric Number of draws from the H-step ahead distribution if Roll = FALSE.
Bands	numeric Vector of probabilities representing the confidence band levels for multi-step ahead parameters forecasts. Only if Roll = FALSE.
ReturnDraws	logical Return the draws from the multi-step ahead predictive distribution when Roll = FALSE ?

Details

Parameter point forecasts retain an H by K matrix structure, including H=1 and K=1. For non-rolling forecasts beyond one step, quantile() and ES() require simulations retained with ReturnDraws = TRUE; otherwise they report an explanatory error. Rolling one-step forecasts and non-rolling one-step forecasts do not require retained simulations for these methods.

Value

An object of the class uGASFor.

Author(s)

Leopoldo Catania

References

Blasques F, Koopman SJ, Lasak K, and Lucas, A (2016). "In-sample Confidence Bands and Out-of-Sample Forecast Bands for Time-Varying Parameters in Observation-Driven Models." International Journal of Forecasting, 32(3), 875-887. doi:10.1016/j.ijforecast.2016.04.002.

Examples

```
# Specify an univariate GAS model with Student-t
# conditional distribution and time-varying location, scale and shape parameter

# Inflation Forecast

set.seed(123)

data("cpichg")
```

```
GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE, scale = TRUE, shape = FALSE))

# Perform H-step ahead forecast with confidence bands

Fit = UniGASFit(GASSpec, cpichg)
Forecast = UniGASFor(Fit, H = 12)

Forecast

# Perform 1-Step ahead rolling forecast

InsampleData = cpichg[1:250]
OutSampleData = cpichg[251:276]

Fit = UniGASFit(GASSpec, InsampleData)

Forecast = UniGASFor(Fit, Roll = TRUE, out = OutSampleData)

Forecast
```

`UniGASRoll` *Rolling forecast with univariate GAS models*

Description

One-step ahead rolling forecasts with model re-estimation. The function also reports several quantity for backtesting for point and density forecasts.

Usage

```
UniGASRoll(data, GASSpec, ForecastLength = 500, Nstart = NULL,
           RefitEvery = 23, RefitWindow = c("moving", "recursive"),
           cluster = NULL, Compute.SE = FALSE, ...)
```

Arguments

`data`	`numeric` vector containing the time series of observations.
`GASSpec`	An object of the class uGASSpec created using the function UniGASSpec.
`ForecastLength`	`numeric` Length of the out-of-sample.
`Nstart`	`numeric` Number of observations in the first estimation sample. Ignored if `ForecastLength` is supplied.
`RefitEvery`	`numeric` Number of periods before model coefficients re-estimation.
`RefitWindow`	`character` Type of window. If `RefitWindow = "recursive"` all the observations are used when the model is re-estimated. If `RefitWindow = "moving"` old observations are eliminated.

cluster	A cluster object created calling using the paralell package. If supplied parallel processing is used to speed up the computations.
Compute.SE	logical. Should asymptotic Standard Errors be computed? By default Compute.SE = FALSE
...	Additional arguments for UniGASFit

Details

Hessian computation is disabled by default. Pass Compute.Hessian = TRUE through ... to enable it for each refit, independently of Compute.SE.

If ForecastLength is explicitly supplied, it determines the forecast horizon and Nstart is ignored. Otherwise, a supplied Nstart is the number of observations in the first estimation sample, and the horizon is the sample size minus Nstart. If neither is supplied, ForecastLength defaults to 500. The resulting forecast horizon must be positive and smaller than the sample size. RefitEvery must be a positive integer.

Value

An object of the class uGASRoll.

Author(s)

Leopoldo Catania

Examples

```
# Specify an univariate GAS model with Student-t
# conditional distribution and time-varying location, scale and shape parameter

# Inflation Forecast

data("cpichg")
help(cpichg)

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE, scale = TRUE, shape = FALSE))

# Perform 1-step ahead rolling forecast with refit
library("parallel")

Roll = UniGASRoll(cpichg, GASSpec, ForecastLength = 50,
                  RefitEvery = 12, RefitWindow = c("moving"))


Roll
```

`UniGASSim` *Simulate Univariate GAS processes*

Description

Simulate Univariate GAS processes.

Usage

```
UniGASSim(fit = NULL, T.sim = 1000,
          kappa = NULL, A = NULL, B = NULL, Dist = NULL, ScalingType = NULL)
```

Arguments

`fit`	An estimated object of the class uGASFit. By default `fit = NULL`.
`T.sim`	`numeric` Length of the simulated time series.
`kappa`	`numeric` Vector of intercepts in the recursion for the reparametrised parameters. For a target state $\bar{f}$, use $\kappa = (I - B)\bar{f}$. Only used if `fit = NULL`
`A`	`matrix` Of coefficients of dimension K x K that premultiply the conditional score in the GAS updating recursion, see Details. Only used if `fit = NULL`
`B`	`matrix` Of autoregressive coefficients of dimension K x K, see Details. Only used if `fit = NULL`
`Dist`	`character` Label of the conditional distribution, see DistInfo. Only used if `fit = NULL`
`ScalingType`	`character` Indicating the scaling mechanism for the conditional score. Possible choices are `"Identity"`, `"Inv"`, `"InvSqrt"`. Note that, for some distribution only `ScalingType = "Identity"` is supported, see the function DistInfo. When `ScalingType = "InvSqrt"` the inverse of the Cholesky decomposition of the information matrix is used. Default value `ScalingType = "Identity"`. Only used if `fit = NULL`

Details

The function permits to simulate from an estimated uGASFit object. If `fit` is not provided, the user can specify a GAS model via the additional arguments `kappa`, `A`, `B`, `Dist` and `ScalingType`.

All the information regarding the supported univariate conditional distributions can be investigated using the DistInfo function. The model is specified as

$$y_t \sim p(y|\theta_t)$$

, where θ_t is the vector of parameters for the density $p(y|.)$. Note that, θ_t includes also those parameters that are not time-varying. The GAS recursion for θ_t is

$$\theta_t = \Lambda(\tilde{\theta}_t)$$

,

$$\tilde{\theta}_t = \kappa + A * s_{t-1} + B * \tilde{\theta}_{t-1}$$

, where $\Lambda(.)$ is the mapping function (see UniMapParameters) and $\tilde{\theta}_t$ is the vector of reparametrised parameters. The process is initialized at $\tilde{\theta}_1 = (I - B)^{-1}\kappa$, where κ is the vKappa vector. The vector s_t is the scaled score of $p(y|.)$ with respect to $\tilde{\theta}_t$. See Ardia et. al. (2016a) for further details.

Value

An object of the class uGASSim.

Author(s)

Leopoldo Catania

References

Ardia D, Boudt K and Catania L (2016a). "Generalized Autoregressive Score Models in R: The GAS Package." https://www.ssrn.com/abstract=2825380.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Examples

```
# Simulate from a GAS process with Student-t conditional
# distribution, time-varying location, scale and fixed shape parameter.

library(GAS)

set.seed(786)

T.sim  = 1000 # number of observations to simulate
Dist   = "std" # conditional Studen-t distribution


# Target initial distribution parameters: location = 0, scale = 1.5,
# degrees of freedom = 7. Use the package's current parameter mapping.
fbar = UniUnmapParameters(c(0, 1.5, 7), Dist)

# in this way we specify that the shape parameter is constant while the score
# coefficients for the location and the scale
# parameters are 0.001 and 0.01, respectively.

A = matrix(c(0.001 , 0.0  , 0.0 ,
             0.0   , 0.01 , 0.0 ,
             0.0   , 0.0  , 0.0 ), 3, byrow = TRUE)
```

```
B = matrix(c(0.7 , 0.0 , 0.0 ,
             0.0 , 0.98, 0.0 ,
             0.0 , 0.0 , 0.0),3,byrow = TRUE) # Matrix of autoregressive parameters.

# Convert the target transformed state into the recursion intercept.
kappa = (diag(length(fbar)) - B) %*% fbar

Sim = UniGASSim(fit = NULL, T.sim, kappa, A, B, Dist, ScalingType = "Identity")

Sim
```

UniGASSpec *Univariate GAS specification*

Description

Specify the conditional distribution, scaling mechanism and time–varying parameters for univariate GAS models.

Usage

```
UniGASSpec(Dist = "norm", ScalingType = "Identity",
           GASPar = list(location = FALSE, scale = TRUE,
                         skewness = FALSE, shape = FALSE, shape2 = FALSE))
```

Arguments

Dist	character Indicating the label of the conditional distribution. Available distribution can be displayed using the function DistInfo. Default valueDist = "norm".
ScalingType	character Indicating the scaling mechanism for the conditional score. Possible choices are "Identity", "Inv", "InvSqrt". Note that, for some distribution only ScalingType = "Identity" is supported, see the function DistInfo. When ScalingType = "InvSqrt" the inverse of the cholesky decomposition of the information matrix is used. Both snorm and sstd support all three mechanisms using their full analytic Fisher information matrices. Default value ScalingType = "Identity".
GASPar	list Containing information about which parameters of the conditional distribution have to be time-varying. location = TRUE refers to the location parameter, scale = TRUE refers to the scale parameter, skewness = TRUE refers to the parameter controlling the skewness, shape = TRUE refers to the shape parameter (e.g. the degree of freedom of the Student-t distribution), shape2 = TRUE refers to the second shape parameter. If the distribution specified in the Dist argument does not have, say, a shape parameter, the condition shape = TRUE or shape = FALSE is ignored. Note that, for some distributions, these labels are not strictly related to their literal statistical meaning. Indeed, for the Exponential distribution exp, the term location indicates the usual intensity rate parameter. See the DistInfo function for more details.

Details

For skellam, location controls the mean-state coordinate $f_1 = \mu$, whereas scale controls $f_2 = \frac{1}{2}\log(v^2 - \mu^2)$. Thus a dynamic mean may change the variance even when scale = FALSE. See UniMapParameters. Coefficients for the old Skellam state parameterization are not interchangeable with these coefficients.

All the information regarding the supported univariate conditional distributions can be investigated using the DistInfo function.

Value

An object of the class uGASSpec.

Author(s)

Leopoldo Catania

References

Ardia D, Boudt K and Catania L (2016). "Generalized Autoregressive Score Models in R: The GAS Package." https://www.ssrn.com/abstract=2825380.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." Journal of Applied Econometrics, 28(5), 777-795. doi:10.1002/jae.1279.

Harvey AC (2013). Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press.

Examples

```
# Specify an univariate GAS model with Student-t
# conditional distribution and time-varying location, scale and shape parameter
library("GAS")

GASSpec = UniGASSpec(Dist = "std", ScalingType = "Identity",
                     GASPar = list(location = TRUE,
                                   scale = TRUE, shape = TRUE))

GASSpec
```

UniMapParameters *Mapping function for univariate distributions*

Description

Map unrestricted vector of parameters into the proper space. This function transforms the parameters updated using the GAS recursion into their proper space.

Usage

```
UniMapParameters(Theta_tilde, Dist)
```

Arguments

`Theta_tilde`	`numeric` Vector of reparametrised parameters, see Details.
`Dist`	`character` Label of the conditional distribution, see DistInfo.

Details

For `Dist = "skellam"`, the public distribution parameters are mean μ and variance $v > |\mu|$. The state coordinates are

$$f_1 = \mu, \qquad f_2 = \tfrac{1}{2}\log(v^2 - \mu^2).$$

The forward map is

$$\mu = f_1, \qquad v = \sqrt{f_1^2 + \exp(2f_2)}.$$

Its Jacobian, with parameter coordinates in rows and state coordinates in columns, is

$$J = \begin{pmatrix} 1 & 0 \\ f_1/v & \exp(2f_2)/v \end{pmatrix}.$$

States that cannot represent a finite interior parameter pair in floating-point arithmetic return `NA`; the coordinates are not clipped independently. Models estimated using the previous Skellam mapping must be refitted.

The order of the parameters is generally: `location`, `scale`, `skewness`, `shape`, `shape2`. When the distribution defined by `Dist` does not have, say, the `shape` parameter, this should be simply omitted. See also DistInfo for specific distributions.

Value

A `numeric` vector of parameters.

Author(s)

Leopoldo Catania

Examples

```
# Map unrestricted parameters for the Student-t distribution.
library("GAS")

Dist = "std"

# Vector of unconditional reparametrised parameters such that,
# Theta = c(0, 1.5 ,7), i.e., location = 0, scale = 1.5,
# degrees of freedom = 7.

# Use the inverse mapping so the distribution-specific bounds match.
Theta_tilde = UniUnmapParameters(c(0, 1.5, 7), Dist)
```

```
Theta = UniMapParameters(Theta_tilde, Dist)

Theta
```

`UniUnmapParameters` *Unmapping function for univariate distributions, i.e. inverse of UniMapParameters*

Description

Transform distribution parameters into the unrestricted parameters. The unrestricted vector of parameters is updated using the GAS recursion.

Usage

```
UniUnmapParameters(Theta, Dist)
```

Arguments

`Theta`	`numeric` Vector of parameters, see Details.
`Dist`	`character` Label of the conditional distribution, see DistInfo.

Details

For `Dist = "skellam"`, the public distribution parameters are mean μ and variance $v > |\mu|$. The state coordinates are

$$f_1 = \mu, \qquad f_2 = \tfrac{1}{2}\log(v^2 - \mu^2).$$

The forward map is

$$\mu = f_1, \qquad v = \sqrt{f_1^2 + \exp(2f_2)}.$$

Its Jacobian, with parameter coordinates in rows and state coordinates in columns, is

$$J = \begin{pmatrix} 1 & 0 \\ f_1/v & \exp(2f_2)/v \end{pmatrix}.$$

States that cannot represent a finite interior parameter pair in floating-point arithmetic return `NA`; the coordinates are not clipped independently. Models estimated using the previous Skellam mapping must be refitted.

The order of the parameters is generally: `location`, `scale`, `skewness`, `shape`, `shape2`. When the distribution defined by `Dist` does not have, say, the `shape` parameter, this should be simply omitted. See also DistInfo for specific distributions.

Value

A `numeric` vector of parameters.

Author(s)

Leopoldo Catania

Examples

```
# Unmap parameters for the Student-t distribution
library("GAS")

Dist = "std"

# Vector of parameters such that,
# Theta = c(0, 1.5 ,7), i.e., location = 0, scale = 1.5,
# degrees of freedom = 7.

Theta = c(0.1, 1.5, 7)

Theta_tilde = UniUnmapParameters(Theta, Dist)

Theta_tilde

# It works.
all(abs(UniMapParameters(Theta_tilde, Dist) - Theta) < 1e-16)
```

---

usunp *US Monthly Civilian Unemployment Rate (UNRATE) from 1948-01-01 to 2016-05-01*

Description

From https://fred.stlouisfed.org/series/UNRATE: The unemployment rate represents the number of unemployed as a percentage of the labor force. Labor force data are restricted to people 16 years of age and older, who currently reside in 1 of the 50 states or the District of Columbia, who do not reside in institutions (e.g., penal and mental facilities, homes for the aged), and who are not on active duty in the Armed Forces.

Usage

```
data("usunp")
```

Format

A xts object containing 821 observations from 1948-01-01 to 2016-05-01.

References

US. Bureau of Labor Statistics, Civilian Unemployment Rate [UNRATE], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/UNRATE, July 2, 2016.

Index

∗ **classes**
  mGASFit, 19
  mGASFor, 20
  mGASRoll, 21
  mGASSim, 22
  mGASSpec, 23
  uGASFit, 44
  uGASFor, 45
  uGASRoll, 46
  uGASSim, 48
  uGASSpec, 49
∗ **datasets**
  cpichg, 10
  dji30ret, 13
  Goals, 18
  sp500ret, 41
  sp500rv, 42
  StockIndices, 42
  tqdata, 43
  usunp, 63
∗ **package**
  GAS-package, 3

BacktestDensity, 4
BacktestVaR, 6
build_mR *(*MultiMapParameters*)*, 34
build_vR *(*MultiMapParameters*)*, 34

coef,mGASFit-method *(*mGASFit*)*, 19
coef,mGASRoll-method *(*mGASRoll*)*, 21
coef,mGASSim-method *(*mGASSim*)*, 22
coef,uGASFit-method *(*uGASFit*)*, 44
coef,uGASRoll-method *(*uGASRoll*)*, 46
coef,uGASSim-method *(*uGASSim*)*, 48
ConfidenceBands, 7
convergence *(*mGASFit*)*, 19
convergence,mGASFit-method *(*mGASFit*)*, 19
convergence,uGASFit-method *(*uGASFit*)*, 44
cpichg, 10

data.frame, *43*
ddist_Multi *(*distributions*)*, 11
ddist_Uni *(*distributions*)*, 11
DistInfo, 10, *12*, *13*, *22*, *30*, *32–35*, *48*, *57*, *59–62*
DistLabels *(*DistInfo*)*, 10
DistName *(*DistInfo*)*, 10
DistParameters *(*DistInfo*)*, 10
distributions, 11
DistScalingType *(*DistInfo*)*, 10
DistType *(*DistInfo*)*, 10
dji30ret, 13

ES *(*uGASFit*)*, 44
ES,uGASFit-method *(*uGASFit*)*, 44
ES,uGASFor-method *(*uGASFor*)*, 45
ES,uGASRoll-method *(*uGASRoll*)*, 46
ES,uGASSim-method *(*uGASSim*)*, 48

fn.optim, 14, *24*, *50*
fn.solnp, 15
function, *14*, *16*
FZLoss, 16

GAS, *40*
GAS *(*GAS-package*)*, 3
GAS-package, 3
getFilteredParameters, *9*
getFilteredParameters *(*uGASFit*)*, 44
getFilteredParameters,mGASFit-method *(*mGASFit*)*, 19
getFilteredParameters,mGASSim-method *(*mGASSim*)*, 22
getFilteredParameters,uGASFit-method *(*uGASFit*)*, 44
getFilteredParameters,uGASSim-method *(*uGASSim*)*, 48
getForecast *(*uGASRoll*)*, 46
getForecast,mGASFor-method *(*mGASFor*)*, 20

getForecast,mGASRoll-method *(mGASRoll)*, 21
getForecast,uGASFor-method *(uGASFor)*, 45
getForecast,uGASRoll-method *(uGASRoll)*, 46
getMoments *(uGASFit)*, 44
getMoments,mGASFit-method *(mGASFit)*, 19
getMoments,mGASFor-method *(mGASFor)*, 20
getMoments,mGASRoll-method *(mGASRoll)*, 21
getMoments,mGASSim-method *(mGASSim)*, 22
getMoments,uGASFit-method *(uGASFit)*, 44
getMoments,uGASFor-method *(uGASFor)*, 45
getMoments,uGASRoll-method *(uGASRoll)*, 46
getMoments,uGASSim-method *(uGASSim)*, 48
getObs *(uGASFit)*, 44
getObs,mGASFit-method *(mGASFit)*, 19
getObs,mGASSim-method *(mGASSim)*, 22
getObs,uGASFit-method *(uGASFit)*, 44
getObs,uGASFor-method *(uGASFor)*, 45
getObs,uGASRoll-method *(uGASRoll)*, 46
getObs,uGASSim-method *(uGASSim)*, 48
Goals, 18

hist, *39*

IM_Uni *(distributions)*, 11

LogScore *(uGASRoll)*, 46
LogScore,mGASFor-method *(mGASFor)*, 20
LogScore,mGASRoll-method *(mGASRoll)*, 21
LogScore,uGASFor-method *(uGASFor)*, 45
LogScore,uGASRoll-method *(uGASRoll)*, 46
LowerA *(NumericalBounds)*, 37
LowerB *(NumericalBounds)*, 37
LowerNu *(NumericalBounds)*, 37

MapR_C *(MultiMapParameters)*, 34
mdist_Uni *(distributions)*, 11
mGASFit, *8*, 19, *25*, *27*, *30*, *40, 41*
mGASFit-class *(mGASFit)*, 19
mGASFor, 20, *27*, *40, 41*
mGASFor-class *(mGASFor)*, 20
mGASMultiForcast *(MultiGASFor)*, 26
mGASRoll, 21, *29*, *40, 41*
mGASRoll-class *(mGASRoll)*, 21
mGASSim, 22, *31*, *40, 41*
mGASSim-class *(mGASSim)*, 22
mGASSpec, *14*, *16*, 23, *24*, *29*, *33*
mGASSpec-class *(mGASSpec)*, 23
MultiGASFit, 23, *27*, *29*
MultiGASFor, 26
MultiGASRoll, *21*, 28
MultiGASSim, 30
MultiGASSpec, *14*, *16*, *24*, *29, 30*, 32
MultiMapParameters, *31*, 34, *35, 36*
MultiUnmapParameters, *34*, 35

NumberParameters *(DistInfo)*, 10
NumericalBounds, 37

optim, *14*, *24*, *50*

pdist_Uni *(distributions)*, 11
pit *(uGASFit)*, 44
pit,uGASFit-method *(uGASFit)*, 44
pit,uGASFor-method *(uGASFor)*, 45
pit,uGASRoll-method *(uGASRoll)*, 46
PIT_test, 38, *45–47*
plot *(plot-methods)*, 40
plot,mGASFit,missing-method *(plot-methods)*, 40
plot,mGASFor,missing-method *(plot-methods)*, 40
plot,mGASRoll,missing-method *(plot-methods)*, 40
plot,mGASSim,missing-method *(plot-methods)*, 40
plot,uGASFit,missing-method *(plot-methods)*, 40
plot,uGASFor,missing-method *(plot-methods)*, 40
plot,uGASRoll,missing-method *(plot-methods)*, 40
plot,uGASSim,missing-method *(plot-methods)*, 40
plot-methods, 40
PlotMenu *(plot-methods)*, 40

qdist_Uni *(distributions)*, 11
quantile,uGASFit-method *(uGASFit)*, 44
quantile,uGASFor-method *(uGASFor)*, 45
quantile,uGASRoll-method *(uGASRoll)*, 46
quantile,uGASSim-method *(uGASSim)*, 48
Quantiles *(distributions)*, 11

rdist_Multi *(distributions)*, 11

rdist_Uni *(distributions)*, 11
residuals *(uGASFit)*, 44
residuals,mGASFit-method *(mGASFit)*, 19
residuals,mGASRoll-method *(mGASRoll)*, 21
residuals,uGASFit-method *(uGASFit)*, 44
residuals,uGASRoll-method *(uGASRoll)*, 46
rmvt_mat *(distributions)*, 11

Score_Multi *(distributions)*, 11
Score_Uni *(distributions)*, 11
show,mGASFit-method *(mGASFit)*, 19
show,mGASFor-method *(mGASFor)*, 20
show,mGASRoll-method *(mGASRoll)*, 21
show,mGASSim-method *(mGASSim)*, 22
show,mGASSpec-method *(mGASSpec)*, 23
show,uGASFit-method *(uGASFit)*, 44
show,uGASFor-method *(uGASFor)*, 45
show,uGASRoll-method *(uGASRoll)*, 46
show,uGASSim-method *(uGASSim)*, 48
show,uGASSpec-method *(uGASSpec)*, 49
solnp, *15, 16*
sp500ret, 41
sp500rv, 42
StockIndices, 42
summary,mGASFit-method *(mGASFit)*, 19
summary,uGASFit-method *(uGASFit)*, 44

tqdata, 43

uGASFit, *8*, *40, 41*, 44, *51*, *54*, *57*
uGASFit-class *(uGASFit)*, 44
uGASFor, *40, 41*, 45, *54*
uGASFor-class *(uGASFor)*, 45
uGASMultiForcast *(UniGASFor)*, 53
uGASRoll, *4*, *40, 41*, 46, *56*
uGASRoll-class *(uGASRoll)*, 46
uGASSim, *40, 41*, 48, *58*
uGASSim-class *(uGASSim)*, 48
uGASSpec, *14*, *16*, *19*, *44*, 49, *50*, *55*, *60*
uGASSpec-class *(uGASSpec)*, 49
UniGASFit, 49, *54*, *56*
UniGASFor, 53
UniGASRoll, *47*, 55
UniGASSim, 57
UniGASSpec, *14*, *16*, *50*, *55*, 59
UniMapParameters, *58*, *60*, 60, *62*
UniUnmapParameters, 62
UnMapR_C *(MultiUnmapParameters)*, 35
UpperA *(NumericalBounds)*, 37
UpperB *(NumericalBounds)*, 37
UpperNu *(NumericalBounds)*, 37
usunp, 63

xts, *10*, *42*, *63*

zoo, *18*