Package 'uGMAR'

June 19, 2025

Title Estimate Univariate Gaussian and Student's t Mixture Autoregressive Models

Version 3.6.0

Author Savi Virolainen [aut, cre] (ORCID: <https://orcid.org/0000-0002-5075-6821>)

Description Maximum likelihood estimation of univariate Gaussian Mixture Autoregressive (GMAR), Student's t Mixture Autoregressive (StMAR), and Gaussian and Student's t Mixture Autoregressive (G-StMAR) models,
quantile residual tests, graphical diagnostics, forecast and simulate from GMAR, StMAR and G-StMAR processes.
Leena Kalliovirta, Mika Meitz, Pentti Saikkonen (2015) <doi:10.1111/jtsa.12108>,
Mika Meitz, Daniel Preve, Pentti Saikkonen (2023) <doi:10.1080/03610926.2021.1916531>,
Savi Virolainen (2022) <doi:10.1515/snde-2020-0060>.

Depends R (>= 3.4.0)

BugReports https://github.com/saviviro/uGMAR/issues

License GPL-3

Encoding UTF-8

LazyData true

Imports Brobdingnag (>= 1.2-4), parallel, pbapply (>= 1.3-2), stats (>= 3.3.2), gsl (>= 1.9-10.3)

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2025-06-19 16:20:02 UTC

Maintainer Savi Virolainen <savi.virolainen@helsinki.fi>

Contents

uGMAR-package *uGMAR: Estimate Univariate Gaussian and Student's t Mixture Autoregressive Models*

Description

uGMAR is a package for estimating univariate Gaussian mixture autoregressive (GMAR), Student's t mixture autoregressive (StMAR), and Gaussian and Student's t mixture autoregressive (G-StMAR) models. In addition to unconstrained and constrained estimation, uGMAR provides tools for quantile residual based model diagnostics, forecasting, simulation, and more.

The readme file or the vignette is a good place to start.

Author(s)

Maintainer: Savi Virolainen <savi.virolainen@helsinki.fi> (ORCID)

See Also

Useful links:

- Report bugs at https://github.com/saviviro/uGMAR/issues

add_data *Add data to object of class 'gsmar' defining a GMAR, StMAR, or G-StMAR model*

Description

add_data adds or updates data to object of class 'gsmar' that defines a GMAR, StMAR, or G-StMAR model. Also calculates empirical mixing weights, conditional moments, and quantile residuals accordingly.

Usage

```
add_data(
  data,
  gsmar,
  calc_qresiduals = TRUE,
  calc_cond_moments = TRUE,
  calc_std_errors = FALSE,
  custom_h = NULL
)
```

Arguments

data	a numeric vector or class 'ts' object containing the data. NA values are not supported.
gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.
calc_qresiduals	should quantile residuals be calculated? Default is TRUE iff the model contains data.
calc_cond_moments	should conditional means and variances be calculated? Default is TRUE iff the model contains data.
calc_std_errors	should approximate standard errors be calculated?
custom_h	A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If NULL (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is 6e-6 for the other parameters.

Value

Returns an object of class 'gsmar' defining the GMAR, StMAR, or G-StMAR model with the data added to the model. If the object already contained data, the data will be updated. Does not modify the 'gsmar' object given as argument!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

fitGSMAR, GSMAR, iterate_more, get_gradient, get_regime_means, swap_parametrization, stmar_to_gstmar

Examples

```
# G-StMAR model without data
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
gstmar42 <- GSMAR(p=4, M=c(1, 1), params=params42gs, model="G-StMAR")
gstmar42

# Add data to the model
```

```
gstmar42 <- add_data(data=M10Y1Y, gsmar=gstmar42)
gstmar42
```

`alt_gsmar` *Construct a GSMAR model based on results from an arbitrary estimation round of* `fitGSMAR`

Description

`alt_gsmar` constructs a GSMAR model based on results from an arbitrary estimation round of `fitGSMAR`.

Usage

```
alt_gsmar(
  gsmar,
  which_round = 1,
  which_largest,
  calc_qresiduals = TRUE,
  calc_cond_moments = TRUE,
  calc_std_errors = TRUE,
  custom_h = NULL
)
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.
`which_round`	based on which estimation round should the model be constructed? An integer value in 1,...,`ncalls`.
`which_largest`	based on estimation round with which largest log-likelihood should the model be constructed? An integer value in 1,...,`ncalls`. For example, `which_largest=2` would take the second largest log-likelihood and construct the model based on the corresponding estimates. If specified, then `which_round` is ignored.
`calc_qresiduals`	should quantile residuals be calculated? Default is `TRUE` iff the model contains data.
`calc_cond_moments`	should conditional means and variances be calculated? Default is `TRUE` iff the model contains data.
`calc_std_errors`	should approximate standard errors be calculated?
`custom_h`	A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If `NULL` (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is `6e-6` for the other parameters.

Details

It's sometimes useful to examine other estimates than the one with the highest log-likelihood value. This function is just a simple wrapper to `GSMAR` that picks the correct estimates from an object returned by `fitGSMAR`.

In addition to the S3 methods listed under the topic "Methods (by generic)", the `predict` and `simulate` methods are also available for the class 'gsmar' objects (see `?predict.gsmar` and `?simulate.gsmar`).

Value

Returns an object of class `'gsmar'` defining the specified GMAR, StMAR, or G-StMAR model. If data is supplied, the returned object contains (by default) empirical mixing weights, some conditional and unconditional moments, and quantile residuals. Note that the first p observations are taken as the initial values so the mixing weights, conditional moments, and quantile residuals start from the p+1:th observation (interpreted as t=1).

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

`fitGSMAR`, `GSMAR`, `iterate_more`, `get_gradient`, `get_regime_means`, `swap_parametrization`, `stmar_to_gstmar`

Examples

```
# These are long running examples that take approximately ...
fit42t <- fitGSMAR(data=M10Y1Y, p=4, M=2, model="StMAR", ncalls=2,
                   seeds=c(1, 6))
fit42t # Bad estimate in the boundary of the stationarity region!

# So we build a model based on the next-best local maximum point:
fit42t_alt <- alt_gsmar(fit42t, which_largest=2)
fit42t_alt # Overly large degrees of freedom paramter estimate

# Switch to the appropriate G-StMAR model:
fit42gs <- stmar_to_gstmar(fit42t_alt)
fit42gs
```

calc_gradient *Calculate gradient or Hessian matrix*

Description

calc_gradient or calc_hessian calculates the gradient or Hessian matrix of the given function at the given point using central difference numerical approximation. get_gradient (and get_foc) or get_hessian calculates the gradient or Hessian matrix of the log-likelihood function at the parameter values of a class 'gsmar' object. get_soc returns eigenvalues of the Hessian matrix.

Usage

```
calc_gradient(x, fn, h = 6e-06, varying_h = NULL, ...)

calc_hessian(x, fn, h = 6e-06, varying_h = NULL, ...)

get_gradient(gsmar, custom_h = NULL)

get_foc(gsmar, custom_h = NULL)

get_hessian(gsmar, custom_h = NULL)

get_soc(gsmar, custom_h = NULL)
```

Arguments

x	a numeric vector specifying the point at which the gradient or Hessian should be evaluated.
fn	a function that takes in the argument x as the **first** argument.
h	the difference used to approximate the derivatives.
varying_h	a numeric vector with the same length as x specifying the difference h for each dimension separately. If NULL (default), then the difference given as parameter h will be used for all dimensions.
...	other arguments passed to fn.
gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.
custom_h	same as varying_h but if NULL (default), then the difference h used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is 6e-6 for the other parameters.

Details

In particular, the functions get_foc and get_soc can be used to check whether the found estimates denote a (local) maximum point, a saddle point, or something else.

Value

The gradient functions return numerical approximation of the gradient, and the Hessian functions return numerical approximation of the Hessian. `get_soc` returns eigenvalues of the Hessian matrix, `get_foc` is the same as `get_gradient` but named conveniently.

Warning

No argument checks!

See Also

`profile_logliks`

Examples

```
# Simple function
foo <- function(x) x^2 + x
calc_gradient(x=1, fn=foo)
calc_gradient(x=-0.5, fn=foo)
calc_hessian(x=2, fn=foo)

# More complicated function
foo <- function(x, a, b) a*x[1]^2 - b*x[2]^2
calc_gradient(x=c(1, 2), fn=foo, a=0.3, b=0.1)
calc_hessian(x=c(1, 2), fn=foo, a=0.3, b=0.1)

# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(data=simudata, p=1, M=2, params=params12, model="GMAR")
get_gradient(gmar12)
get_foc(gmar12)
get_hessian(gmar12)
get_soc(gmar12)
```

---

cond_moments *Calculate conditional moments of GMAR, StMAR, or G-StMAR model*

Description

`cond_moments` calculates the regime specific conditional means and variances and total conditional means and variances of the specified GMAR, StMAR or G-StMAR model.

Usage

```
cond_moments(
  data,
  p,
  M,
```

```
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  parametrization = c("intercept", "mean"),
  to_return = c("regime_cmeans", "regime_cvars", "total_cmeans", "total_cvars")
)
```

Arguments

`data`	a numeric vector or class `'ts'` object containing the data. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive order of the model.
`M`	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.
`params`	a real valued parameter vector specifying the model. **For non-restricted models:** Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where • υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$,σ_m^2) • $\boldsymbol{\phi}_m$= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$ • ν= (ν_{M1+1}, ..., ν_M) • $M1$ is the number of GMAR type regimes. In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$. If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors $\boldsymbol{\phi}_m$ with the vectors $\boldsymbol{\psi}_m$ that satisfy $\boldsymbol{\phi}_m$=$C_m\boldsymbol{\psi}_m$ (see the argument `constraints`). **For restricted models:** Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where $\boldsymbol{\phi}$=(ϕ_1, ..., ϕ_p) contains the AR coefficients, which are common for all regimes. If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector $\boldsymbol{\phi}$ with the vector $\boldsymbol{\psi}$ that satisfies $\boldsymbol{\phi}$=$C\boldsymbol{\psi}$ (see the argument `constraints`). Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.
`model`	is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`parametrization`

is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`to_return` calculate regimewise conditional means (`regime_cmeans`), regimewise conditional variances (`regime_cvars`), total conditional means (`total_cmeans`), or total conditional variances (`total_cvars`)?

Value

Note that the first p observations are taken as the initial values so the conditional moments start form the p+1:th observation (interpreted as t=1).

if `to_return=="regime_cmeans"`: a size $((n_obs - p) \times M)$ matrix containing the regime specific conditional means.

if `to_return=="regime_cvars"`: a size $((n_obs - p) \times M)$ matrix containing the regime specific conditional variances.

if `to_return=="total_cmeans"`: a size $((n_obs - p) \times 1)$ vector containing the total conditional means.

if `to_return=="total_cvars"`: a size $((n_obs - p) \times 1)$ vector containing the total conditional variances.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

Other moment functions: `get_regime_autocovs()`, `get_regime_means()`, `get_regime_vars()`, `uncond_moments()`

Examples

```
# GMAR model, regimewise conditional means and variances
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
cond_moments(simudata, p=1, M=2, params=params12, model="GMAR",
             to_return="regime_cmeans")
cond_moments(simudata, p=1, M=2, params=params12, model="GMAR",
             to_return="regime_cvars")

# G-StMAR-model, total conditional means and variances
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
cond_moments(M10Y1Y, p=4, M=c(1, 1), params=params42gs, model="G-StMAR",
             to_return="total_cmeans")
cond_moments(M10Y1Y, p=4, M=c(1, 1), params=params42gs, model="G-StMAR",
             to_return="total_cvars")
```

---

cond_moment_plot *Conditional mean or variance plot for GMAR, StMAR, and G-StMAR models*

Description

`cond_moment_plot` plots the one-step in-sample conditional means/variances of the model along with the time series contained in the model (e.g. the time series the model was fitted to). Also plots the regimewise conditional means/variances multiplied with the mixing weights.

Usage

```
cond_moment_plot(gsmar, which_moment = c("mean", "variance"))
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.
`which_moment`	should conditional means or variances be plotted?

Details

The conditional mean plot works best if the data contains positive values only.

Value

`cond_moment_plot` only plots to a graphical device and does not return anything. Numerical values of the conditional means/variances can be extracted from the model with the dollar sign.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

profile_logliks, diagnostic_plot, fitGSMAR, GSMAR, quantile_residual_tests, quantile_residual_plot

Examples

```
# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(data=simudata, p=1, M=2, params=params12, model="GMAR")
cond_moment_plot(gmar12, which_moment="mean")
cond_moment_plot(gmar12, which_moment="variance")

# G-StMAR model
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
gstmar42 <- GSMAR(data=M10Y1Y, p=4, M=c(1, 1), params=params42gs,
                  model="G-StMAR")
cond_moment_plot(gstmar42, which_moment="mean")
cond_moment_plot(gstmar42, which_moment="variance")
```

---

diagnostic_plot *Quantile residual based diagnostic plots for GMAR, StMAR, and G-StMAR models*

Description

diagnostic_plot plots quantile residual time series, normal QQ-plot, autocorrelation function, and squared quantile residual autocorrelation function. There is an option to also plot the individual statistics associated with the quantile residual tests (for autocorrelation and conditional heteroskedasticity) divided by their approximate standard errors with their approximate 95% critical bounds (see Kalliovirta 2012, Section 3).

Usage

```
diagnostic_plot(gsmar, nlags = 20, nsimu = 1, plot_indstats = FALSE)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.
nlags	a positive integer specifying how many lags should be calculated for the autocorrelation and conditional heteroscedasticity statistics.
nsimu	a positive integer specifying to how many simulated values from the process the covariance matrix "Omega" (used to compute the tests) should be based on. Larger number of simulations may result more reliable tests but takes longer to compute. If smaller than data size, then "Omega" will be based on the given data. Ignored if plot_indstats==FALSE.
plot_indstats	set TRUE if the individual statistics discussed in Kalliovirta (2012) should be plotted with their approximate 95% critical bounds (this may take some time).

Details

Sometimes the individual statistics are not plotted because it was not (numerically) possible to calculate all the required statistics. This may suggest that the model is misspecified.

The dashed lines plotted with autocorrelation functions (for quantile residuals and their squares) are plus-minus $1.96 * T^{-1/2}$ where T is the sample size (minus the p initial values for conditional models).

Value

diagnostic_plot only plots to a graphical device and does not return anything. Use the function quantile_residual_tests in order to obtain the individual statistics.

Suggested packages

Install the suggested package "gsl" for faster evaluations in the cases of StMAR and G-StMAR models. For large StMAR and G-StMAR models with large data the calculations to obtain the individual statistics may take a significantly long time without the package "gsl".

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

profile_logliks, get_foc, fitGSMAR, cond_moment_plot, quantile_residual_tests, quantile_residual_plot, simulate.gsmar, LR_test, Wald_test

Examples

```
## The below examples the approximately 30 seconds to run.

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)
diagnostic_plot(fit42gs)

# Restricted StMAR model: plot also the individual statistics with
# their approximate critical bounds using the given data (and not
# simulation procedure)
fit42tr <- fitGSMAR(M10Y1Y, p=4, M=2, model="StMAR", restricted=TRUE,
                    ncalls=1, seeds=1)
diagnostic_plot(fit42tr, nlags=10, nsimu=1, plot_indstats=TRUE)

# GMAR model, plot 30 lags.
fit12 <- fitGSMAR(data=simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)
diagnostic_plot(fit12, nlags=30)
```

`fitGSMAR` *Estimate Gaussian or Student's t Mixture Autoregressive model*

Description

`fitGSMAR` estimates GMAR, StMAR, or G-StMAR model in two phases. In the first phase, a genetic algorithm is employed to find starting values for a gradient based method. In the second phase, the gradient based variable metric algorithm is utilized to accurately converge to a local maximum or a saddle point near each starting value. Parallel computing is used to conduct multiple rounds of estimations in parallel.

Usage

```
fitGSMAR(
  data,
  p,
  M,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  ncalls = round(10 + 9 * log(sum(M))),
  ncores = 2,
  maxit = 500,
  seeds = NULL,
  print_res = TRUE,
  filter_estimates = TRUE,
```

```
  ...
)
```

Arguments

`data`	a numeric vector or class `'ts'` object containing the data. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive order of the model.
`M`	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.
`model`	is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.
`restricted`	a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.
`constraints`	specifies linear constraints imposed to each regime's autoregressive parameters separately. **For non-restricted models:** a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$. **For restricted models:** a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$. The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.
`conditional`	a logical argument specifying whether the conditional or exact log-likelihood function should be used.
`parametrization`	is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?
`ncalls`	a positive integer specifying how many rounds of estimation should be conducted. The estimation results may vary from round to round because of multimodality of the log-likelihood function and the randomness associated with the genetic algorithm.
`ncores`	the number of CPU cores to be used in the estimation process.
`maxit`	the maximum number of iterations for the variable metric algorithm.
`seeds`	a length `ncalls` vector containing the random number generator seed for each call to the genetic algorithm, or `NULL` for not initializing the seed. Exists for the purpose of creating reproducible results.
`print_res`	should the estimation results be printed?

`filter_estimates`
: should likely inappropriate estimates be filtered? See details.

`...`
: additional settings passed to the function `GAfit` employing the genetic algorithm.

Details

Because of complexity and multimodality of the log-likelihood function, it's **not guaranteed** that the estimation algorithm will end up in the global maximum point. It's often expected that most of the estimation rounds will end up in some local maximum point instead, and therefore a number of estimation rounds is required for reliable results. Because of the nature of the models, the estimation may fail particularly in the cases where the number of mixture components is chosen too large. Note that the genetic algorithm is designed to avoid solutions with mixing weights of some regimes too close to zero at almost all times ("redundant regimes") but the settings can, however, be adjusted (see ?GAfit).

If the iteration limit for the variable metric algorithm (`maxit`) is reached, one can continue the estimation by iterating more with the function `iterate_more`.

The core of the genetic algorithm is mostly based on the description by *Dorsey and Mayer (1995)*. It utilizes a slightly modified version the individually adaptive crossover and mutation rates described by *Patnaik and Srinivas (1994)* and employs (50%) fitness inheritance discussed by *Smith, Dike and Stegmann (1995)*. Large (in absolute value) but stationary AR parameter values are generated with the algorithm proposed by Monahan (1984).

The variable metric algorithm (or quasi-Newton method, Nash (1990, algorithm 21)) used in the second phase is implemented with function the `optim` from the package `stats`.

Additional Notes about the estimates:

Sometimes the found MLE is very close to the boundary of the stationarity region some regime, the related variance parameter is very small, and the associated mixing weights are "spiky". This kind of estimates often maximize the log-likelihood function for a technical reason that induces by the endogenously determined mixing weights. In such cases, it might be more appropriate to consider the next-best local maximum point of the log-likelihood function that is well inside the parameter space. Models based local-only maximum points can be built with the function `alt_gsmar` by adjusting the argument `which_largest` accordingly.

Some mixture components of the StMAR model may sometimes get very large estimates for the degrees of freedom parameters. Such parameters are weakly identified and induce various numerical problems. However, mixture components with large degree of freedom parameter estimates are similar to the mixture components of the GMAR model. It's hence advisable to further estimate a G-StMAR model by allowing the mixture components with large degrees of freedom parameter estimates to be GMAR type with the function `stmar_to_gstmar`.

Filtering inappropriate estimates: If `filter_estimates == TRUE`, the function will automatically filter out estimates that it deems "inappropriate". That is, estimates that are not likely solutions of interest. Specifically, it filters out solutions that incorporate regimes with any modulus of the roots of the AR polynomial less than 1.0015; a variance parameter estimat near zero (less than 0.0015); mixing weights such that they are close to zero for almost all t for at least one regime; or mixing weight parameter estimate close to zero (or one). You can also set `filter_estimates=FALSE` and find the solutions of interest yourself by using the function `alt_gsmar`.

Value

Returns an object of class 'gsmar' defining the estimated GMAR, StMAR or G-StMAR model. The returned object contains estimated mixing weights, some conditional and unconditional moments, and quantile residuals. Note that the first p observations are taken as the initial values, so the mixing weights, conditional moments, and quantile residuals start from the p+1:th observation (interpreted as t=1). In addition, the returned object contains the estimates and log-likelihoods from all of the estimation rounds. See ?GSMAR for the form of the parameter vector, if needed.

S3 methods

The following S3 methods are supported for class 'gsmar' objects: print, summary, plot, predict, simulate, logLik, residuals.

References

- Dorsey R. E. and Mayer W. J. 1995. Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, **13**, 53-66.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Monahan J.F. 1984. A Note on Enforcing Stationarity in Autoregressive-Moving Average Models. *Biometrica* **71**, 403-404.
- Nash J. 1990. Compact Numerical Methods for Computers. Linear algebra and Function Minimization. *Adam Hilger*.
- Patnaik L.M. and Srinivas M. 1994. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *Transactions on Systems, Man and Cybernetics* **24**, 656-667.
- Smith R.E., Dike B.A., Stegmann S.A. 1995. Fitness inheritance in genetic algorithms. *Proceedings of the 1995 ACM Symposium on Applied Computing*, 345-350.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

GSMAR, iterate_more, , stmar_to_gstmar, add_data, profile_logliks, swap_parametrization, get_gradient, simulate.gsmar, predict.gsmar, diagnostic_plot, quantile_residual_tests, cond_moments, uncond_moments, LR_test, Wald_test

Examples

```
## These are long running examples that use parallel computing.
## The below examples take approximately 90 seconds to run.

## Note that the number of estimation rounds (ncalls) is relatively small
## in the below examples to reduce the time required for running the examples.
## For reliable results, a large number of estimation rounds is recommended!
```

```
# GMAR model
fit12 <- fitGSMAR(data=simudata, p=1, M=2, model="GMAR", ncalls=4, seeds=1:4)
summary(fit12)
plot(fit12)
profile_logliks(fit12)
diagnostic_plot(fit12)

# StMAR model (large estimate of the degrees of freedom)
fit42t <- fitGSMAR(data=M10Y1Y, p=4, M=2, model="StMAR", ncalls=2, seeds=c(1, 6))
summary(fit42t) # Overly large 2nd regime degrees of freedom estimate!
fit42gs <- stmar_to_gstmar(fit42t) # Switch to G-StMAR model
summary(fit42gs) # An appropriate G-StMVAR model with one G and one t regime
plot(fit42gs)

# Restricted StMAR model
fit42r <- fitGSMAR(M10Y1Y, p=4, M=2, model="StMAR", restricted=TRUE,
                   ncalls=2, seeds=1:2)
fit42r

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)
fit42gs

# The following three examples demonstrate how to apply linear constraints
# to the autoregressive (AR) parameters.

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
C22 <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(M10Y1Y, p=2, M=2, constraints=C22, ncalls=1, seeds=6)
fit22c

# StMAR(3, 1) model with the second order AR coefficient constrained to zero.
C31 <- list(matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
fit31tc <- fitGSMAR(M10Y1Y, p=3, M=1, model="StMAR", constraints=C31,
                    ncalls=1, seeds=1)
fit31tc

# Such StMAR(3, 2) model that the AR coefficients are restricted to be
# the same for both regimes and the second AR coefficients are
# constrained to zero.
fit32rc <- fitGSMAR(M10Y1Y, p=3, M=2, model="StMAR", restricted=TRUE,
                    constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2),
                    ncalls=1, seeds=1)
fit32rc
```

GAfit *Genetic algorithm for preliminary estimation of GMAR, StMAR, or G-StMAR model*

Description

GAfit estimates specified GMAR, StMAR, or G-StMAR model using a genetic algorithm. The employed genetic algorithm is designed to find starting values for gradient based methods.

Usage

```
GAfit(
  data,
  p,
  M,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  parametrization = c("intercept", "mean"),
  conditional = TRUE,
  ngen = 200,
  popsize,
  smart_mu = min(100, ceiling(0.5 * ngen)),
  mu_scale,
  sigma_scale,
  initpop = NULL,
  regime_force_scale = 1,
  red_criteria = c(0.05, 0.01),
  to_return = c("alt_ind", "best_ind"),
  minval,
  seed = NULL,
  ...
)
```

Arguments

data	a numeric vector or class 'ts' object containing the data. NA values are not supported.
p	a positive integer specifying the autoregressive order of the model.
M	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components M1 in the first element and *StMAR type* components M2 in the second element. The total number of mixture components is M=M1+M2.
model	is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first M1 components are *GMAR type* and the rest M2 components are *StMAR type*.
restricted	a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.
constraints	specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`parametrization`

is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used.

`ngen` a positive integer specifying the number of generations to be ran through in the genetic algorithm.

`popsize` a positive even integer specifying the population size in the genetic algorithm. Default is `10*d` where `d` is the number of parameters.

`smart_mu` a positive integer specifying the generation after which the random mutations in the genetic algorithm are "smart". This means that mutating individuals will mostly mutate fairly close (or partially close) to the best fitting individual so far.

`mu_scale` a real valued vector of length two specifying the mean (the first element) and standard deviation (the second element) of the normal distribution from which the μ_m mean-parameters are generated in random mutations in the genetic algorithm. Default is `c(mean(data), sd(data))`. Note that the genetic algorithm optimizes with mean-parametrization even when `parametrization=="intercept"`, but input (in `initpop`) and output (return value) parameter vectors may be intercept-parametrized.

`sigma_scale` a positive real number specifying the standard deviation of the (zero mean, positive only by taking absolute value) normal distribution from which the component variance parameters are generated in the random mutations in the genetic algorithm. Default is `var(stats::ar(data, order.max=10)$resid, na.rm=TRUE)`.

`initpop` a list of parameter vectors from which the initial population of the genetic algorithm will be generated from. The parameter vectors should be of form...

For non-restricted models: Size $(M(p+3)+M-M1-1\times1)$ vector θ=$(\upsilon_1, ...,\upsilon_M, \alpha_1, ..., \alpha_{M-1},\nu)$ where

- υ_m= $(\phi_{m,0},\phi_m,\sigma_m^2)$
- ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p}), m = 1, ..., M$
- ν= $(\nu_{M1+1}, ..., \nu_M)$
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector $\theta=(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,...,\alpha_{M-1},\nu)$, where $\phi=(\phi_1,...,\phi_p)$ contains the AR coefficients, which are common for all regimes.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies $\phi=C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m=\phi_{m,0}/(1-\sum\phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2. If not specified (or `NULL` as is default), the initial population will be drawn randomly.

`regime_force_scale`
: a non-negative real number specifying how much should natural selection favor individuals with less regimes that have almost all mixing weights (practically) at zero (see `red_criteria`), i.e., with less "redundant regimes". Set to zero for no favoring or large number for heavy favoring. Without any favoring the genetic algorithm gets more often stuck in an area of the parameter space where some regimes are wasted, but with too much favoring the best genes might never mix into the population and the algorithm might converge poorly. Default is `1` and it gives $2x$ larger surviving probability weights for individuals with no wasted regimes compared to individuals with one wasted regime. Number `2` would give $3x$ larger probabilities etc.

`red_criteria`
: a length 2 numeric vector specifying the criteria that is used to determine whether a regime is redundant or not. Any regime `m` which satisfies `sum(mixing_weights[,m] > red_criteria[1]) < red_criteria[2]*n_obs` will be considered "redundant". One should be careful when adjusting this argument (set `c(0, 0)` to fully disable the 'redundant regime' features from the algorithm).

`to_return`
: should the genetic algorithm return the best fitting individual which has the least "redundant" regimes (`"alt_ind"`) or the individual which has the highest log-likelihood in general (`"best_ind"`) but might have more wasted regimes?

`minval`
: a real number defining the minimum value of the log-likelihood function that will be considered. Values smaller than this will be treated as they were `minval` and the corresponding individuals will never survive. The default is `-(10^(ceiling(log10(length(data + 1) - 1)`, and one should be very careful if adjusting this.

`seed`
: a single value, interpreted as an integer, or NULL, that sets seed for the random number generator in the beginning of the function call. If calling `GAfit` from `fitGSMAR`, use the argument `seeds` instead of passing the argument `seed`.

`...`
: We currently use this to catch deprecated arguments.

Details

The core of the genetic algorithm is mostly based on the description by *Dorsey and Mayer (1995)*. It utilizes a slightly modified version of the individually adaptive crossover and mutation rates described by *Patnaik and Srinivas (1994)* and employs (50%) fitness inheritance discussed by *Smith,*

Dike and Stegmann (1995). Large (in absolute value) but stationary AR parameter values are generated with the algorithm proposed by Monahan (1984).

By "redundant" or "wasted" regimes we mean regimes that have the time varying mixing weights basically at zero for all t. The model with redundant regimes would have approximately the same log-likelihood value without the redundant regimes and there is no purpose to have redundant regimes in the model.

Value

Returns estimated parameter vector with the form described in `initpop`.

References

- Dorsey R. E. and Mayer W. J. 1995. Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, **13**, 53-66.
  - Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
  - Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
  - Monahan J.F. 1984. A Note on Enforcing Stationarity in Autoregressive-Moving Average Models. *Biometrica* **71**, 403-404.
  - Patnaik L.M. and Srinivas M. 1994. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *Transactions on Systems, Man and Cybernetics* **24**, 656-667.
  - Smith R.E., Dike B.A., Stegmann S.A. 1995. Fitness inheritance in genetic algorithms. *Proceedings of the 1995 ACM Symposium on Applied Computing*, 345-350.
  - Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

Examples

```
## These are long running examples

# Preliminary estimation of GMAR p=1, M=2, model with the genetic algorithm
# using only 100 generations (200 is recommended):
pars12_ga <- GAfit(data=simudata, p=1, M=2, model="GMAR", ngen=100, seed=1)
pars12_ga # Returns a parameter vector, not a class 'gsmar' object.
```

`get_ar_roots` *Calculate absolute values of the roots of the AR characteristic polynomials*

Description

`get_ar_roots` calculates the absolute values of the roots of the AR characteristic polynomials for each mixture component.

Usage

```
get_ar_roots(gsmar)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.

Value

Returns a list with `M` elements each containing the absolute values of the roots of the AR characteristic polynomial corresponding to each mixture component.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

Examples

```
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(data=simudata, p=1, M=2, params=params12, model="GMAR")
get_ar_roots(gmar12)
```

---

get_regime_autocovs *Calculate regime specific autocovariances* γ_m, p

---

Description

`get_regime_autocovs` calculates the first p regime specific autocovariances $\gamma_{m,p}$ for the given GMAR, StMAR, or G-StMAR model.

Usage

```
get_regime_autocovs(gsmar)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.

Value

Returns a size (pxM) matrix containing the first p autocovariances of the components processes: i:th autocovariance in the i:th row and m:th component process in the m:th column.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: `cond_moments()`, `get_regime_means()`, `get_regime_vars()`, `uncond_moments()`

Examples

```
# GMAR model
params13 <- c(1.4, 0.88, 0.26, 2.46, 0.82, 0.74, 5.0, 0.68, 5.2, 0.72, 0.2)
gmar13 <- GSMAR(p=1, M=3, params=params13, model="GMAR")
get_regime_autocovs(gmar13)

# StMAR model
params12t <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 100, 3.6)
stmar12t <- GSMAR(p=1, M=2, params=params12t, model="StMAR")
get_regime_autocovs(stmar12t)

# G-StMAR model (similar to the StMAR model above)
params12gs <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 3.6)
gstmar12 <- GSMAR(p=1, M=c(1, 1), params=params12gs, model="G-StMAR")
get_regime_autocovs(gstmar12)
```

`get_regime_means` *Calculate regime specific means μ_m*

Description

`get_regime_means` calculates the regime means $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$ for the given GMAR, StMAR, or G-StMAR model

Usage

```
get_regime_means(gsmar)
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.

Value

Returns a length M vector containing the regime mean μ_m in the m:th element.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

cond_moments, uncond_moments, get_regime_vars, get_regime_autocovs

Other moment functions: cond_moments(), get_regime_autocovs(), get_regime_vars(), uncond_moments()

Examples

```
# GMAR model
params13 <- c(1.4, 0.88, 0.26, 2.46, 0.82, 0.74, 5.0, 0.68, 5.2, 0.72, 0.2)
gmar13 <- GSMAR(p=1, M=3, params=params13, model="GMAR")
get_regime_means(gmar13)

# StMAR model
params12t <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 100, 3.6)
stmar12t <- GSMAR(p=1, M=2, params=params12t, model="StMAR")
get_regime_means(stmar12t)

# G-StMAR model (similar to the StMAR model above)
params12gs <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 3.6)
gstmar12 <- GSMAR(p=1, M=c(1, 1), params=params12gs, model="G-StMAR")
get_regime_means(gstmar12)
```

---

get_regime_vars *Calculate regime specific variances* $\gamma\_m, 0$

Description

get_regime_vars calculates the unconditional regime specific variances $\gamma_{m,0}$ for the given GMAR, StMAR, or G-StMAR model.

Usage

```
get_regime_vars(gsmar)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.

Value

Returns a length M vector containing the unconditional variances of the components processes: m:th element for the m:th regime.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: cond_moments(), get_regime_autocovs(), get_regime_means(), uncond_moments()

Examples

```
# GMAR model
params13 <- c(1.4, 0.88, 0.26, 2.46, 0.82, 0.74, 5.0, 0.68, 5.2, 0.72, 0.2)
gmar13 <- GSMAR(p=1, M=3, params=params13, model="GMAR")
get_regime_vars(gmar13)

# StMAR model
params12t <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 100, 3.6)
stmar12t <- GSMAR(p=1, M=2, params=params12t, model="StMAR")
get_regime_vars(stmar12t)

# G-StMAR model (similar to the StMAR model above)
params12gs <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 3.6)
gstmar12 <- GSMAR(p=1, M=c(1, 1), params=params12gs, model="G-StMAR")
get_regime_vars(gstmar12)
```

GSMAR — *Create object of class 'gsmar' defining a GMAR, StMAR, or G-StMAR model*

Description

GSMAR creates an S3 object of class 'gsmar' that defines a GMAR, StMAR, or G-StMAR model.

Usage

```
GSMAR(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  calc_qresiduals,
  calc_cond_moments,
  calc_std_errors = FALSE,
  custom_h = NULL
)

## S3 method for class 'gsmar'
logLik(object, ...)

## S3 method for class 'gsmar'
residuals(object, ...)

## S3 method for class 'gsmar'
summary(object, ..., digits = 2)

## S3 method for class 'gsmar'
plot(x, ..., include_dens = TRUE)

## S3 method for class 'gsmar'
print(x, ..., digits = 2, summary_print = FALSE)
```

Arguments

data	a numeric vector or class 'ts' object containing the data. NA values are not supported.
p	a positive integer specifying the autoregressive order of the model.
M	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components M1 in the first element and *StMAR type* components M2 in the second element. The total number of mixture components is M=M1+M2.
params	a real valued parameter vector specifying the model. **For non-restricted models:** Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where • υ_m= ($\phi_{m,0}$,ϕ_m,σ_m^2)

- ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p}), m = 1, ..., M$
- ν= $(\nu_{M1+1}, ..., \nu_M)$
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2$,$\alpha_1, ..., \alpha_{M-1}$,$\nu$), where ϕ=$(\phi_1, ..., \phi_p)$ contains the AR coefficients, which are common for all regimes.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used.

`parametrization`
is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`calc_qresiduals`
should quantile residuals be calculated? Default is `TRUE` iff the model contains data.

`calc_cond_moments`
: should conditional means and variances be calculated? Default is `TRUE` iff the model contains data.

`calc_std_errors`
: should approximate standard errors be calculated?

`custom_h`
: A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If `NULL` (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is `6e-6` for the other parameters.

`object`
: object of class `'gsmar'` created with `fitGSMAR` or `GSMAR`.

`...`
: in the plot method: arguments passed to the function `density` which calculates the kernel density estimate of the data.

`digits`
: number of digits to be printed (max 20)

`x`
: object of class `'gsmar'` created with `fitGSMAR` or `GSMAR`.

`include_dens`
: Plot also kernel density estimate of the data and model implied stationary density with regimewise densities? See the details.

`summary_print`
: if set to `TRUE` then the print will include approximate standard errors for the estimates, log-likelihood, information criteria values, modulus of the roots of the characteristic AR polynomials for each regime, and several unconditional moments.

Details

Models can be built without data, e.q., in order to simulate from the process, but some things such as quantile residuals and conditional moments can't be calculated without data.

If `include_dens == TRUE`, the kernel density estimate of the data is calculated with the function `density` from the package `stats`. By default, the default settings of that function are used, including the usage of Gaussian kernel. Use the dot parameters to adjust the settings if desired.

By the model implied stationary density we mean the stationary one-dimensional mixture density of M regimes (see KMS 2015, Theorem 1 and the discussion following it for the Gaussian case and Theorem 2 in PMS 2018 for the Student's t case). The regimewise densities (i.e. each density 1,...,M in the stationary mixture density) are multiplied with the mixing weight parameters accordingly.

In the density plot black represents the kernel density estimate of the data, grey dashed line the model implied density, and the colored dotted lines the regime wise densities.

Value

Returns an object of class `'gsmar'` defining the specified GMAR, StMAR, or G-StMAR model. If data is supplied, the returned object contains (by default) empirical mixing weights, some conditional and unconditional moments, and quantile residuals. Note that the first p observations are taken as the initial values so the mixing weights, conditional moments, and quantile residuals start from the p+1:th observation (interpreted as t=1).

Functions

- `logLik(gsmar)`: Log-likelihood method
- `residuals(gsmar)`: residuals method to extract quantile residuals
- `summary(gsmar)`: summary method, standard errors in brackets
- `plot(gsmar)`: Plot method for class 'gsmar'
- `print(gsmar)`: print method

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

`fitGSMAR`, `iterate_more`, `add_data`, `stmar_to_gstmar`, `swap_parametrization`, `get_gradient`, `simulate.gsmar`, `predict.gsmar`, `cond_moments`, `uncond_moments`, `LR_test`, `Wald_test`

Examples

```
# GMAR model without data
params22 <- c(0.9, 0.4, 0.2, 0.5, 0.7, 0.5, -0.2, 0.7, 0.7)
gmar22 <- GSMAR(p=2, M=2, params=params22, model="GMAR")
gmar22

# StMAR model, without data
params12t <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 300, 3.6)
stmar12t <- GSMAR(p=1, M=2, params=params12t, model="StMAR")
stmar12t

# G-StMAR model with data
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
gstmar42 <- GSMAR(data=M10Y1Y, p=4, M=c(1, 1), params=params42gs,
                  model="G-StMAR")
gstmar42

# Restricted G-StMAR model with data
params42gsr <- c(0.13, 0.03, 1.29, -0.4, 0.25, -0.2, 0.03, 0.05, 0.51, 2.76)
gstmar42r <- GSMAR(data=M10Y1Y, p=4, M=c(1, 1), params=params42gsr,
                   model="G-StMAR", restricted=TRUE)
gstmar42r
```

`is_stationary` *Check the stationary condition of specified GMAR, StMAR, or G-StMAR model.*

Description

`is_stationary` checks the stationarity condition of the specified GMAR, StMAR, or G-StMAR model.

Usage

```
is_stationary(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL
)
```

Arguments

`p` a positive integer specifying the autoregressive order of the model.

`M` **For GMAR and StMAR models:** a positive integer specifying the number of mixture components.

For G-StMAR models: a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.

`params` a real valued parameter vector specifying the model.

For non-restricted models: Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where

- υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$,σ_m^2)
- $\boldsymbol{\phi}_m$= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$
- ν= (ν_{M1+1}, ..., ν_M)
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors $\boldsymbol{\phi}_m$ with the vectors $\boldsymbol{\psi}_m$ that satisfy $\boldsymbol{\phi}_m$=$C_m\boldsymbol{\psi}_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where $\boldsymbol{\phi}$=($\boldsymbol{\phi}_1$, ..., $\boldsymbol{\phi}_p$) contains the AR coefficients, which are common for all regimes.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

Details

This function falsely returns `FALSE` for stationary models when the parameter is extremely close to the boundary of the stationarity region.

Value

Returns `TRUE` or `FALSE` accordingly.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

Examples

```
# GMAR model
params22 <- c(0.4, 0.39, 0.6, 0.3, 0.4, 0.1, 0.6, 0.3, 0.8)
is_stationary(p=2, M=2, params=params22)

# StMAR model
params12t <- c(-0.3, 1, 0.9, 0.1, 0.8, 0.6, 0.7, 10, 12)
is_stationary(p=1, M=2, params=params12t, model="StMAR")

# G-StMAR model
params12gs <- c(1, 0.1, 1, 2, 0.2, 2, 0.8, 20)
is_stationary(p=1, M=c(1, 1), params=params12gs, model="G-StMAR")

# Restricted GMAR model
params13r <- c(0.1, 0.2, 0.3, -0.99, 0.1, 0.2, 0.3, 0.5, 0.3)
is_stationary(p=1, M=3, params=params13r, restricted=TRUE)

# Such StMAR(3, 2) that the AR coefficients are restricted to be the
# same for both regimes and that the second AR coefficients are
# constrained to zero.
params32trc <- c(1, 2, 0.8, -0.3, 1, 2, 0.7, 11, 12)
is_stationary(p=3, M=2, params=params32trc, model="StMAR", restricted=TRUE,
              constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
```

---

iterate_more *Maximum likelihood estimation of GMAR, StMAR, or G-StMAR model with preliminary estimates*

Description

`iterate_more` uses a variable metric algorithm to finalize maximum likelihood estimation of a GMAR, StMAR or G-StMAR model (object of class `'gsmar'`) which already has preliminary estimates.

Usage

```
iterate_more(gsmar, maxit = 100, custom_h = NULL, calc_std_errors = TRUE)
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.
`maxit`	the maximum number of iterations for the variable metric algorithm.
`custom_h`	A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If `NULL` (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is `6e-6` for the other parameters.

`calc_std_errors`
: should approximate standard errors be calculated?

Details

The main purpose of `iterate_more` is to provide a simple and convenient tool to finalize the estimation when the maximum number of iterations is reached when estimating a model with the main estimation function `fitGSMAR`. `iterate_more` is essentially a wrapper for the functions `optim` from the package `stats` and `GSMAR` from the package `uGMAR`.

Value

Returns an object of class `'gsmar'` defining the estimated model.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

`fitGSMAR`, `GSMAR`, `stmar_to_gstmar`, `profile_logliks`, `optim`

Examples

```
# Estimate GMAR model with on only 1 iteration in variable metric algorithm
fit12 <- fitGSMAR(simudata, p=1, M=2, maxit=1, ncalls=1, seeds=1)
fit12

# Iterate more since iteration limit was reached
fit12 <- iterate_more(fit12)
fit12
```

---

loglikelihood *Compute the log-likelihood of GMAR, StMAR, or G-StMAR model*

Description

`loglikelihood` computes the log-likelihood of the specified GMAR, StMAR, or G-StMAR model. Exists for convenience if one wants to for example plot profile log-likelihoods or employ other estimation algorithms. Use `minval` to control what happens when the parameter vector is outside the parameter space.

Usage

```
loglikelihood(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  return_terms = FALSE,
  minval = NA
)
```

Arguments

`data` a numeric vector or class `'ts'` object containing the data. `NA` values are not supported.

`p` a positive integer specifying the autoregressive order of the model.

`M` **For GMAR and StMAR models:** a positive integer specifying the number of mixture components.

For G-StMAR models: a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.

`params` a real valued parameter vector specifying the model.

For non-restricted models: Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where

- υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$,σ_m^2)
- $\boldsymbol{\phi}_m$= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$
- ν= (ν_{M1+1}, ..., ν_M)
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors $\boldsymbol{\phi}_m$ with the vectors $\boldsymbol{\psi}_m$ that satisfy $\boldsymbol{\phi}_m$=$C_m\boldsymbol{\psi}_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where $\boldsymbol{\phi}$=(ϕ_1, ..., ϕ_p) contains the AR coefficients, which are common for all regimes.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector $\boldsymbol{\phi}$ with the vector $\boldsymbol{\psi}$ that satisfies $\boldsymbol{\phi}$=$C\boldsymbol{\psi}$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace

each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used.

`parametrization` is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`return_terms` should the terms $l_t : t = 1, .., T$ in the log-likelihood function (see *KMS 2015, eq.(13)* or MPS 2018, eq.(15)) be returned instead of the log-likelihood value?

`minval` this will be returned when the parameter vector is outside the parameter space and `boundaries==TRUE`.

Value

Returns the log-likelihood value or the terms described in `return_terms`.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

fitGSMAR, GSMAR, quantile_residuals, mixing_weights, calc_gradient

Examples

```
# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
loglikelihood(simudata, p=1, M=2, params=params12, model="GMAR")

# G-StMAR-model
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
loglikelihood(M10Y1Y, p=4, M=c(1, 1), params=params42gs, model="G-StMAR")
```

LR_test *Perform likelihood ratio test*

Description

LR_test performs a likelihood ratio test for a GMAR, StMAR, or G-StMAR model.

Usage

```
LR_test(gsmar1, gsmar2)
```

Arguments

gsmar1	an object of class 'gsmar' generated by fitGSMAR or GSMAR, containing the **unconstrained** model.
gsmar2	an object of class 'gsmar' generated by fitGSMAR or GSMAR, containing the **constrained** model.

Details

Performs a likelihood ratio test, testing the null hypothesis that the true parameter value lies in the constrained parameter space specified by constraints imposed to the model gsmar2. Under the null, the test statistic is asymptotically χ^2-distributed with k degrees of freedom, k being the difference in the dimensions of the unconstrained and constrained parameter spaces.

Note that this function does **not** verify that the two models are actually nested. Notably, GSMAR models with different autoregressive orders are not nested, whereas testing models with different numbers of regimes induce an identification problem and thereby unreliable test results (see the discussion related to Theorem 2 in Virolainen, 2021).

Value

A list with class "htest" containing the following components:

`statistic`	the value of the likelihood ratio statistics.
`parameter`	the degrees of freedom of the likelihood ratio statistic.
`p.value`	the p-value of the test.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating the type of the test (likelihood ratio test).
`data.name`	a character string giving the names of the supplied models, gsmar1 and gsmar2.
`gsmar1`	the supplied argument gsmar1
`gsmar2`	the supplied argument gsmar2

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

`Wald_test`, `fitGSMAR`, `GSMAR`, `diagnostic_plot`, `profile_logliks`, `quantile_residual_tests`, `cond_moment_plot`

Examples

```
# GMAR p=1, M=2 model:
fit12 <- fitGSMAR(simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)

# GMAR p=1, M=2 model with AR parameters restricted to be the same in both
# regimes:
fit12r <- fitGSMAR(simudata, p=1, M=2, model="GMAR", restricted=TRUE,
                   ncalls=1, seeds=1)

# Test with likelihood ratio test whether the AR parameters are the same in
# both regimes:
LR_test(fit12, fit12r)
```

M10Y1Y *Spread between 10-Year and 1-Year Treasury rates: M10Y1Y*

Description

A dataset containing monthly U.S. interest rate spread between the 10-Year Treasury constant maturity rate and 1-Year Treasury constant maturity rate from 1982 January to 2020 December.

Usage

```
M10Y1Y
```

Format

A class 'ts' time series object containing 468 observations.

Source

https://fred.stlouisfed.org/series/GS10 https://fred.stlouisfed.org/series/GS1

mixing_weights *Calculate mixing weights of GMAR, StMAR or G-StMAR model*

Description

`mixing_weights` calculates the mixing weights of the specified GMAR, StMAR or G-StMAR model and returns them as a matrix.

Usage

```
mixing_weights(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  parametrization = c("intercept", "mean")
)
```

Arguments

`data` a numeric vector or class `'ts'` object containing the data. `NA` values are not supported.

`p` a positive integer specifying the autoregressive order of the model.

`M` **For GMAR and StMAR models:** a positive integer specifying the number of mixture components.

For G-StMAR models: a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.

`params` a real valued parameter vector specifying the model.

For non-restricted models: Size $(M(p+3)+M-M1-1\times 1)$ vector $\theta=(\upsilon_1, ..., \upsilon_M, \alpha_1, ..., \alpha_{M-1}, \nu)$ where

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m^2)$
- $\phi_m = (\phi_{m,1}, ..., \phi_{m,p}), m = 1, ..., M$
- $\nu = (\nu_{M1+1}, ..., \nu_M)$
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy $\phi_m = C_m\psi_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times 1)$ vector $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}, \nu)$, where $\phi=(\phi_1, ..., \phi_p)$ contains the AR coefficients, which are common for all regimes.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies $\phi = C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying $\phi_m = C_m\psi_m$ for all $m = 1, ..., M$, where $\phi_m = (\phi_{m,1}, ..., \phi_{m,p})$ and $\psi_m = (\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always p for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`parametrization`

is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

Details

The first p observations are taken to be the initial values.

Value

If `to_return=="mw"`**:** a size $((n_obs - p) \times M)$ matrix containing the mixing weights: for m:th component in m:th column.

If `to_return=="mw_tplus1"`**:** a size $((n_obs - p + 1) \times M)$ matrix containing the mixing weights: for m:th component in m:th column. The last row is for $\alpha_{m,T+1}$.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

Examples

```
# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
mixing_weights(simudata, p=1, M=2, params=params12, model="GMAR")

# G-StMAR-model
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
mixing_weights(M10Y1Y, p=4, M=c(1, 1), params=params42gs, model="G-StMAR")
```

pick_pars *Pick ϕ_0 (or μ), AR-coefficients, and variance parameters from a parameter vector*

Description

pick_pars picks ϕ_0/μ, AR-coefficients, and variance parameters from the given parameter vector.

Usage

```
pick_pars(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL
)
```

Arguments

p — a positive integer specifying the autoregressive order of the model.

M — **For GMAR and StMAR models:** a positive integer specifying the number of mixture components.

For G-StMAR models: a size (2×1) integer vector specifying the number of *GMAR type* components M1 in the first element and *StMAR type* components M2 in the second element. The total number of mixture components is M=M1+M2.

params — a real valued parameter vector specifying the model.

For non-restricted models: Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where

- υ_m= ($\phi_{m,0}$,ϕ_m,σ_m^2)
- ϕ_m= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$
- ν= (ν_{M1+1}, ..., ν_M)
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument constraints).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where ϕ=(ϕ_1, ..., ϕ_p) contains the AR coefficients, which are common for all regimes.

If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument constraints).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

Value

Returns a $((p + 2) \times M)$ matrix containing the parameters, column for each component. The first row for ϕ_0 or μ depending on the parametrization, the second row for ϕ_1, ..., the second to last row for ϕ_p, and the last row for σ^2. @keywords internal

---

`plot.gsmarpred` *Plot method for class 'gsmarpred' objects*

Description

`plot.gsmarpred` is plot method for class 'gsmarpred' objects

Usage

```
## S3 method for class 'gsmarpred'
plot(x, ..., nt, mix_weights = TRUE, add_grid = TRUE)
```

Arguments

x	object of class 'gsmarpred' created with predict.gsmar.
...	arguments passed to function grid.
nt	a positive integer specifying the number of observations to be plotted along with the prediction. Default is round(length(data)*0.15).
mix_weights	TRUE if forecasts for mixing weights should be plotted, FALSE in not.
add_grid	should grid a be added to the plots?

Details

This method is intended for plotting forecasts of GSMAR processes.

---

plot.qrtest *Quantile residual tests for GMAR, StMAR , and G-StMAR models*

Description

quantile_residual_tests performs quantile residual tests for GMAR, StMAR, and G-StMAR models, testing normality, autocorrelation, and conditional heteroscedasticity of the quantile residuals.

Usage

```
## S3 method for class 'qrtest'
plot(x, ...)

## S3 method for class 'qrtest'
print(x, ..., digits = 3)

quantile_residual_tests(
  gsmar,
  lags_ac = c(1, 3, 6, 12),
  lags_ch = lags_ac,
  nsimu = 1,
  print_res = TRUE
)
```

Arguments

x	object of class 'qrtest' created with the function quantile_residual_tests.
...	graphical parameters passed to segments in plot.qrtest. Currently not used in print.qrtest
digits	the number of digits to be print
gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.

`lags_ac`	a numeric vector of positive integers specifying the lags for which autocorrelation is tested.
`lags_ch`	a numeric vector of positive integers specifying the lags for which conditional heteroscedasticity is tested.
`nsimu`	a positive integer specifying to how many simulated observations the covariance matrix Omega (see Kalliovirta (2012)) should be based on. If smaller than data size, then omega will be based on the given data and not on simulated data. Having the covariance matrix omega based on a large simulated sample might improve the tests size properties.
`print_res`	a logical argument defining whether the results should be printed or not.

Details

For a correctly specified GSMAR model employing the maximum likelihood estimator, the quantile residuals are asymptotically independent with standard normal distribution. They can hence be used in a similar manner to conventional Pearson's residuals. For more details about quantile residual based diagnostics, and in particular, about the quantile residual tests, see the cited article by *Kalliovirta (2012)*.

Value

Returns an object of class `'qrtest'` containing the test results in data frames. In the cases of autocorrelation and conditional heteroscedasticity tests, the returned object also contains the associated individual statistics and their standard errors, discussed in *Kalliovirta (2012)* at the pages 369-370.

Functions

- `plot(qrtest)`: Plot p-values of the autocorrelation and conditional heteroskedasticity tests.
- `print(qrtest)`: Print method for class 'qrtest' objects

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

`profile_logliks`, `fitGSMAR`, `GSMAR`, `diagnostic_plot`, `predict.gsmar`, `get_test_Omega`,

Examples

```
## The below examples take approximately 30 seconds to run.

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(data=M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)

# Tests based on the observed data (without simulation procedure) with the
# default lags:
qrt1 <- quantile_residual_tests(fit42gs)

# Tests based on the simulation procedure using sample size 10000 and with
# the lags specified by hand:
set.seed(1)
qrt2 <- quantile_residual_tests(fit42gs, lags_ac=c(1, 6), nsimu=10000)

# GMAR model
fit12 <- fitGSMAR(data=simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)
qrt3 <- quantile_residual_tests(fit12, lags_ac=c(1, 5, 10, 15))
```

`predict.gsmar` *Forecast GMAR, StMAR, or G-StMAR process*

Description

`predict.gsmar` forecasts the specified GMAR, StMAR, or G-StMAR process by using the given data to simulate its possible future values. For one-step forecasts using the exact formula for conditional mean is supported.

Usage

```
## S3 method for class 'gsmar'
predict(
  object,
  ...,
  n_ahead,
  nsimu = 10000,
  pi = c(0.95, 0.8),
  pred_type = c("median", "mean", "cond_mean"),
  pi_type = c("two-sided", "upper", "lower", "none"),
  plot_res = TRUE,
  mix_weights = TRUE,
  nt
)
```

Arguments

object	object of class 'gsmar' created with function fitGSMAR or GSMAR.
...	additional arguments passed to grid (ignored if plot_res==FALSE).
n_ahead	a positive integer specifying how many steps in the future should be forecasted.
nsimu	a positive integer specifying to how many simulations the forecast should be based on.
pi	a numeric vector specifying confidence levels for the prediction intervals.
pred_type	should the prediction be based on sample "median" or "mean"? Or should it be one-step-ahead forecast based on the exact conditional mean ("cond_mean")? prediction intervals won't be calculated if the exact conditional mean is used.
pi_type	should the prediction intervals be "two-sided", "upper", or "lower"?
plot_res	a logical argument defining whether the forecast should be plotted or not.
mix_weights	TRUE if forecasts for mixing weights should be plotted, FALSE in not.
nt	a positive integer specifying the number of observations to be plotted along with the prediction. Default is round(length(data)*0.15).

Details

predict.gsmar uses the last p values of the data to simulate nsimu possible future values for each step-ahead. The point prediction is then obtained by calculating the sample median or mean for each step and the prediction intervals are obtained from the empirical fractiles.

The function simulate.gsmar can also be used directly for quantile based forecasting.

Value

Returns a class 'gsmarpred' object containing, among the specifications,...

$pred	Point forecasts
$pred_ints	Prediction intervals
$mix_pred	Point forecasts for mixing weights
mix_pred_ints	Individual prediction intervals for mixing weights, as [, , m], m=1,..,M.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

simulate.gsmar, cond_moments, fitGSMAR, GSMAR, quantile_residual_tests, diagnostic_plot

Examples

```
## These examples take approximately 30 seconds to run.

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)

# Forecast 12 steps ahead based on 10000 simulated sample paths, prediction
# interval confidence levels 0.95 and 0.8, prediction based on sample median,
# and two-sided prediction intevals:
mypred <- predict(fit42gs, n_ahead=12, nsimu=10000, pi=c(0.95, 0.8),
                  pred_type="median", pi_type="two-sided")
mypred
plot(mypred)

# Forecast 24 steps ahead based on 1000 simulated sample paths, prediction
# interval confidence level 0.99 and 0.9, prediction based on sample mean,
# and upper prediction intevals:
mypred2 <- predict(fit42gs, n_ahead=24, nsimu=1000, pi=c(0.99, 0.9),
                   pred_type="mean", pi_type="upper")

# Forecast 24 steps ahead based on 1000 simulated sample paths, prediction
# interval confidence level 0.99, 0.95, 0.9 and 0.8, prediction based on
# sample median, and lower prediction intevals:
mypred3 <- predict(fit42gs, n_ahead=24, nsimu=1000, pi=c(0.99, 0.95, 0.9, 0.8),
                   pred_type="median", pi_type="lower")

# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(data=simudata, p=1, M=2, params=params12, model="GMAR")
pred12 <- predict(gmar12, n_ahead=10, nsimu=1000, pi=c(0.95, 0.9, 0.8),
                  pred_type="median", pi_type="two-sided")
pred12
plot(pred12)

# One-step prediction based on the exact conditional mean:
predict(gmar12, n_ahead=1, pred_type="cond_mean", plot_res=FALSE)
```

---

print.gsmarpred *Print method for class 'gsmarpred' objects*

Description

print.gsmarpred is a print method for call 'gsmarpred' objects created with predict.gsmar.

Usage

```
## S3 method for class 'gsmarpred'
print(x, ..., digits = 2)
```

Arguments

x	object of class 'gsmarpred' generated by predict.gsmar.
...	currently not in use.
digits	the number of digits to be printed

---

print.gsmarsum *Print method from objects of class 'gsmarsum'*

Description

print.gsmarsum is a print method for objects of class 'gsmarsum' created with the summary method summary.gsmar. Approximate standard errors are printed in brackets.

Usage

```
## S3 method for class 'gsmarsum'
print(x, ..., digits)
```

Arguments

x	object of class 'gsmarsum' generated by summary.gsmar.
...	currently not in use.
digits	the number of digits to be printed

---

profile_logliks *Plot profile log-likelihoods around the estimates*

Description

profile_logliks plots profile log-likelihoods around the estimates.

Usage

```
profile_logliks(gsmar, scale = 0.02, nrows, ncols, precision = 200)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.
scale	a numeric scalar specifying the interval plotted for each estimate: the estimate plus-minus abs(scale*estimate).
nrows	how many rows should be in the plot-matrix? The default is max(ceiling(log2(nparams) - 1), 1).
ncols	how many columns should be in the plot-matrix? The default is ceiling(nparams/nrows). Note that nrows*ncols should not be smaller than the number of parameters.
precision	at how many points should each profile log-likelihood be evaluated at?

Details

The red vertical line points the estimate.

Be aware that the profile log-likelihood function is subject to a numerical error due to limited float-point precision when considering extremely large parameter values, say, overly large degrees freedom estimates.

Value

Only plots to a graphical device and doesn't return anything.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

quantile_residual_plot, diagnostic_plot, cond_moment_plot, GSMAR, quantile_residual_tests, simulate.gsmar

Examples

```
## The below examples the approximately 15 seconds to run.

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)
profile_logliks(fit42gs)
```

```
# GMAR model, graphs zoomed in closer.
fit12 <- fitGSMAR(data=simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)
profile_logliks(fit12, scale=0.001)
```

---

quantile_residuals *Compute quantile residuals of GMAR, StMAR, or G-StMAR model*

Description

quantile_residuals computes the quantile residuals of the specified GMAR, StMAR, or G-StMAR model.

Usage

```
quantile_residuals(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  parametrization = c("intercept", "mean")
)
```

Arguments

data	a numeric vector or class 'ts' object containing the data. NA values are not supported.
p	a positive integer specifying the autoregressive order of the model.
M	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components M1 in the first element and *StMAR type* components M2 in the second element. The total number of mixture components is M=M1+M2.
params	a real valued parameter vector specifying the model. **For non-restricted models:** Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where • υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$,σ_m^2) • $\boldsymbol{\phi}_m$= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$ • ν= (ν_{M1+1}, ..., ν_M) • $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}, \nu)$, where ϕ=$(\phi_1, ..., \phi_p)$ contains the AR coefficients, which are common for all regimes.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`parametrization` is the model parametrized with the "intercepts" $\phi_{m,0}$ or "means" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

Details

Numerical integration is employed if the quantile residuals cannot be obtained analytically with the hypergeometric function using the package 'gsl'.

Value

Returns a $(Tx1)$ numeric vector containing the quantile residuals of the specified GMAR, StMAR or G-StMAR model. Note that there are no quantile residuals for the first `p` observations as they are the initial values.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

Examples

```
# GMAR model
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
quantile_residuals(simudata, p=1, M=2, params=params12, model="GMAR")

# G-StMAR-model
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
quantile_residuals(M10Y1Y, p=4, M=c(1, 1), params=params42gs, model="G-StMAR")
```

---

quantile_residual_plot

Plot quantile residual time series and histogram

Description

`quantile_residualsPlot` plots quantile residual time series and histogram.

Usage

```
quantile_residual_plot(gsmar)
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.

Value

Only plots to a graphical device and doesn't return anything.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

profile_logliks, diagnostic_plot, fitGSMAR, GSMAR, quantile_residual_tests, simulate.gsmar

Examples

```
## The below examples the approximately 15 seconds to run.

# G-StMAR model with one GMAR type and one StMAR type regime
fit42gs <- fitGSMAR(M10Y1Y, p=4, M=c(1, 1), model="G-StMAR",
                    ncalls=1, seeds=4)
quantile_residual_plot(fit42gs)

# GMAR model
fit12 <- fitGSMAR(data=simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)
quantile_residual_plot(fit12)
```

---

random_ind — *Create random GMAR, StMAR, or G-StMAR model compatible parameter vector*

Description

random_ind creates a random GMAR, StMAR, or G-StMAR model compatible mean-parametrized parameter vector.

smart_ind creates a random GMAR, StMAR, or G-StMAR model compatible parameter vector close to argument params. Sometimes returns exactly the given parameter vector.

Usage

```
random_ind(
  p,
  M,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  mu_scale,
  sigma_scale,
  forcestat = FALSE
)

smart_ind(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  mu_scale,
  sigma_scale,
  accuracy,
  which_random = numeric(0),
  forcestat = FALSE
)
```

Arguments

`p`	a positive integer specifying the autoregressive order of the model.
`M`	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.
`model`	is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.
`restricted`	a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.
`constraints`	specifies linear constraints imposed to each regime's autoregressive parameters separately. **For non-restricted models:** a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$. **For restricted models:** a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always p for all regimes. Ignore or set to NULL if applying linear constraints is not desired.

mu_scale — a real valued vector of length two specifying the mean (the first element) and standard deviation (the second element) of the normal distribution from which the $\phi_{m,0}$ **or** μ_m (depending on the desired parametrization) parameters (for random regimes) should be generated.

sigma_scale — a positive real number specifying the standard deviation of the (zero mean, positive only by taking absolute value) normal distribution from which the component variance parameters (for random regimes) should be generated.

forcestat — use the algorithm by Monahan (1984) to force stationarity on the AR parameters (slower) for random regimes? Not supported for constrained models.

params — a real valued parameter vector specifying the model.

For non-restricted models: Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where

- υ_m= ($\phi_{m,0}$,ϕ_m,σ_m^2)
- ϕ_m= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$
- ν= (ν_{M1+1}, ..., ν_M)
- $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument constraints).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where ϕ=(ϕ_1, ..., ϕ_p) contains the AR coefficients, which are common for all regimes.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument constraints).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If parametrization=="mean", just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first M1 components are *GMAR type* and the rest M2 components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

accuracy — a real number larger than zero specifying how close to params the generated parameter vector should be. Standard deviation of the normal distribution from which new parameter values are drawn from will be corresponding parameter value divided by accuracy.

which_random — a numeric vector of maximum length M specifying which regimes should be random instead of "smart" when using smart_ind. Does not affect mixing weight parameters. Default in none.

Details

These functions can be used, for example, to create initial populations for the genetic algorithm. Mean-parametrization (instead of intercept terms $\phi_{m,0}$) is assumed.

Value

Returns estimated parameter vector with the form described in `initpop`.

References

- Monahan J.F. 1984. A Note on Enforcing Stationarity in Autoregressive-Moving Average Models. *Biometrica* **71**, 403-404.

Examples

```
set.seed(1)

# GMAR model parameter vector
params22 <- random_ind(p=2, M=2, mu_scale=c(0, 1), sigma_scale=1)
smart22 <- smart_ind(p=2, M=2, params22, accuracy=10)
cbind(params22, smart22)

# Restricted GMAR parameter vector
params12r <- random_ind(p=1, M=2, restricted=TRUE, mu_scale=c(-2, 2), sigma_scale=2)
smart12r <- smart_ind(p=1, M=2, params12r, restricted=TRUE, accuracy=20)
cbind(params12r, smart12r)

# StMAR parameter vector: first regime is random in the "smart individual"
params13t <- random_ind(p=1, M=3, model="StMAR", mu_scale=c(3, 1), sigma_scale=3)
smart13t <- smart_ind(p=1, M=3, params13t, model="StMAR", accuracy=15,
                      mu_scale=c(3, 3), sigma_scale=3, which_random=1)
cbind(params13t, smart13t)

# Restricted StMAR parameter vector
params22tr <- random_ind(p=2, M=2, model="StMAR", restricted=TRUE,
                         mu_scale=c(3, 2), sigma_scale=0.5)
smart22tr <- smart_ind(p=2, M=2, params22tr, model="StMAR", restricted=TRUE,
                       accuracy=30)
cbind(params22tr, smart22tr)

# G-StMAR parameter vector
params12gs <- random_ind(p=1, M=c(1, 1), model="G-StMAR", mu_scale=c(0, 1),
                         sigma_scale=1)
smart12gs <- smart_ind(p=1, M=c(1, 1), params12gs, model="G-StMAR",
                       accuracy=20)
cbind(params12gs, smart12gs)

# Such StMAR(3,2) that the AR coefficients are restricted to be
# the same for both regimes and that the second AR coefficients are
# constrained to zero. Second regime is random in the "smart individual".
params32trc <- random_ind(p=3, M=2, model="StMAR", restricted=TRUE,
                          constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2),
```

```
                          mu_scale=c(-2, 0.5), sigma_scale=4)
smart32trc <- smart_ind(p=3, M=2, params32trc, model="StMAR", restricted=TRUE,
                        constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2),
                        mu_scale=c(0, 0.1), sigma_scale=0.1, which_random=2,
                        accuracy=20)
cbind(params32trc, smart32trc)
```

`reform_parameters` *Reform any parameter vector into standard form.*

Description

`reform_parameters` takes a parameter vector of any (non-constrained) GMAR, StMAR, or G-StMAR model and returns a list with the parameter vector in the standard form, parameter matrix containing AR coefficients and component variances, mixing weights alphas, and in case of StMAR or G-StMAR model also degrees of freedom parameters.

Usage

```
reform_parameters(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE
)
```

Arguments

`p`	a positive integer specifying the autoregressive order of the model.
`M`	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.
`params`	a real valued parameter vector specifying the model. **For non-restricted models:** Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$,$\nu$) where • υ_m= ($\phi_{m,0}$,ϕ_m,σ_m^2) • ϕ_m= ($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$ • ν= ($\nu_{M1+1}, ..., \nu_M$) • $M1$ is the number of GMAR type regimes.

In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors ϕ_m with the vectors ψ_m that satisfy ϕ_m=$C_m\psi_m$ (see the argument `constraints`).

For restricted models: Size $(3M+M-M1+p-1\times1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}, \nu)$, where ϕ=$(\phi_1, ..., \phi_p)$ contains the AR coefficients, which are common for all regimes.
If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector ϕ with the vector ψ that satisfies ϕ=$C\psi$ (see the argument `constraints`).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$.
In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

Details

This function does not support models imposing linear constraints. No argument checks in this function.

Value

Returns a list with...

`$params` parameter vector in the standard form.

`$pars` corresponding parameter matrix containing AR coefficients and component variances. First row for phi0 or means depending on the parametrization. Column for each component.

`$alphas` numeric vector containing mixing weight parameters for all of the components (also for the last one).

`$dfs` numeric vector containing degrees of freedom parameters for all of components. Returned only if `model == "StMAR"` or `model == "G-StMAR"`.

@keywords internal

simudata *Simulated data*

Description

A dataset containing 200 observations simulated from a GMAR p=1, M=2 process.

Usage

```
simudata
```

Format

A numeric vector of length 200.

Source

Simulated

simulate.gsmar *Simulate obsercations from GMAR, StMAR, and G-StMAR processes*

Description

`simulate.gsmar` simulates observations from the specified GMAR, StMAR, or G-StMAR process. Can be utilized for forecasting future values of the process.

Usage

```
## S3 method for class 'gsmar'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  ...,
  init_values = NULL,
  ntimes = 1,
  drop = TRUE
)
```

Arguments

object	object of class 'gsmar', typically created with the function fitGSMAR or GSMAR.
nsim	a positive integer specifying how many values (ahead from init_values) will be simulated.
seed	an integer that specifies the seed for the random number generator. Ignored if NULL.
...	currently not in use.
init_values	a numeric vector with length >=p specifying the initial values for the simulation. The **last** element will be used as the initial value for the first lag, the second last element will be initial value for the second lag, etc. If NULL, initial values will be simulated from the process's stationary distribution.
ntimes	a positive integer specifying how many sets of simulations should be performed.
drop	if TRUE (default) then the components of the returned list are coerced to lower dimension if ntimes==1, i.e., $sample and $component will be vectors and $mixing_weights will be matrix.

Details

The argument ntimes is intended for forecasting: a GSMAR process can be forecasted by simulating its possible future values. One can perform a large number of sets of simulations and calculate the sample quantiles from the simulated values to obtain prediction intervals. See the forecasting example below for a hand-on demonstration.

Value

If drop==TRUE and ntimes==1 (default): $sample and $component are vectors and $mixing_weights is a (nsim$\times M$) matrix. Otherwise, returns a list with...

$sample a size (nsim$\times$ntimes) matrix containing the simulated values.

$component a size (nsim$\times$ntimes) matrix containing the information from which mixture component each value was generated from.

$mixing_weights a size (nsim$\times M\times$ntimes) array containing the mixing weights corresponding to the sample: the dimension [i, ,] is the time index, the dimension [, i,] indicates the regime, and the dimension [, , i] indicates the i:th set of simulations.

References

- Galbraith, R., Galbraith, J. 1974. On the inverses of some patterned matrices arising in the theory of stationary time series. *Journal of Applied Probability* **11**, 63-71.
- Kalliovirta L. (2012) Misspecification tests based on quantile residuals. *The Econometrics Journal*, **15**, 358-393.
- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

fitGSMAR, GSMAR, predict.gsmar, add_data, cond_moments, mixing_weights

Examples

```
set.seed(1)

# GMAR model:
params22 <- c(0.9, 0.4, 0.2, 0.5, 0.7, 0.5, -0.2, 0.7, 0.7)
mod22 <- GSMAR(p=2, M=2, params=params22, model="GMAR")
mysim <- simulate(mod22, nsim=500)
ts.plot(mysim$sample)
ts.plot(mysim$component)
ts.plot(mysim$mixing_weights, col=rainbow(2), lty=2)


# G-StMAR model, with initial values:
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
gstmar42 <- GSMAR(data=M10Y1Y, p=4, M=c(1, 1), params=params42gs,
                  model="G-StMAR")
sim42gs <- simulate(gstmar42, nsim=500, init_values=1:4)
ts.plot(sim42gs$sample)
ts.plot(sim42gs$component)
ts.plot(sim42gs$mixing_weights, col=rainbow(2), lty=2)


# FORECASTING EXAMPLE:
# GMAR model, 1000 sets of simulations with initial values from the data:
params12 <- c(1.70, 0.85, 0.30, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(data=simudata, p=1, M=2, params=params12, model="GMAR")
sim12 <- simulate(gmar12, nsim=5, init_val=gmar12$data, ntimes=1000)
apply(sim12$sample, MARGIN=1, FUN=median) # Point prediction
apply(sim12$sample, MARGIN=1, FUN=quantile, probs=c(0.025, 0.975)) # 95% pi
apply(sim12$mixing_weights, MARGIN=1:2, FUN=median) # mix.weight point pred
apply(sim12$mixing_weights, MARGIN=1:2, FUN=quantile,
      probs=c(0.025, 0.975)) # mix.weight 95% prediction intervals
```

`stmarpars_to_gstmar` *Transform a StMAR or G-StMAR model parameter vector to a corresponding G-StMAR model parameter vector with large dfs parameters reduced.*

Description

`stmarpars_to_gstmar` transforms a StMAR model parameter vector to a corresponding G-StMAR model parameter vector with large dfs parameters reduced by switching the related regimes to be GMAR type.

Usage

```
stmarpars_to_gstmar(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  maxdf = 100
)
```

Arguments

p	a positive integer specifying the autoregressive order of the model.
M	**For GMAR and StMAR models:** a positive integer specifying the number of mixture components. **For G-StMAR models:** a size (2×1) integer vector specifying the number of *GMAR type* components `M1` in the first element and *StMAR type* components `M2` in the second element. The total number of mixture components is `M=M1+M2`.
params	a real valued parameter vector specifying the model. **For non-restricted models:** Size $(M(p+3)+M-M1-1\times1)$ vector θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1},ν) where • υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$,σ_m^2) • $\boldsymbol{\phi}_m$= ($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$ • ν= (ν_{M1+1}, ..., ν_M) • $M1$ is the number of GMAR type regimes. In the **GMAR** model, $M1 = M$ and the parameter ν dropped. In the **StMAR** model, $M1 = 0$. If the model imposes **linear constraints** on the autoregressive parameters: Replace the vectors $\boldsymbol{\phi}_m$ with the vectors $\boldsymbol{\psi}_m$ that satisfy $\boldsymbol{\phi}_m$=$C_m\boldsymbol{\psi}_m$ (see the argument `constraints`). **For restricted models:** Size $(3M+M-M1+p-1\times1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2,α_1, ..., α_{M-1},ν), where $\boldsymbol{\phi}$=($\boldsymbol{\phi}_1$, ..., $\boldsymbol{\phi}_p$) contains the AR coefficients, which are common for all regimes. If the model imposes **linear constraints** on the autoregressive parameters: Replace the vector $\boldsymbol{\phi}$ with the vector $\boldsymbol{\psi}$ that satisfies $\boldsymbol{\phi}$=$C\boldsymbol{\psi}$ (see the argument `constraints`). Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. If `parametrization=="mean"`, just replace each intercept term $\phi_{m,0}$ with the regimewise mean $\mu_m = \phi_{m,0}/(1 - \sum \phi_{i,m})$. In the **G-StMAR** model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*. Note that in the case **M=1**, the mixing weight parameters α are dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν have to be larger than 2.

`model` is "GMAR", "StMAR", or "G-StMAR" model considered? In the G-StMAR model, the first `M1` components are *GMAR type* and the rest `M2` components are *StMAR type*.

`restricted` a logical argument stating whether the AR coefficients $\phi_{m,1}, ..., \phi_{m,p}$ are restricted to be the same for all regimes.

`constraints` specifies linear constraints imposed to each regime's autoregressive parameters separately.

For non-restricted models: a list of size $(p \times q_m)$ constraint matrices C_m of full column rank satisfying ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ϕ_m= $(\phi_{m,1}, ..., \phi_{m,p})$ and ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: a size $(p \times q)$ constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

The symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is not desired.

`maxdf` regimes with degrees of freedom parameter value larger than this will be turned into GMAR type.

Value

Returns a list with three elements: `$params` contains the corresponding G-StMAR model parameter vector, `$reg_order` contains the permutation that was applied to the regimes (GMAR type components first, and decreasing ordering by mixing weight parameters), and `$M` a vector of length two containing the number of GMAR type regimes in the first element and the number of StMAR type regimes in the second.

Examples

```
params12 <- c(2, 0.9, 0.1, 0.8, 0.5, 0.5, 0.4, 12, 300)
stmarpars_to_gstmar(p=1, M=2, params=params12, model="StMAR", maxdf=100)
```

`stmar_to_gstmar` *Estimate a G-StMAR model based on a StMAR model with large degrees of freedom parameters*

Description

`stmar_to_gstmar` estimates a G-StMAR model based on a StMAR model with large degree of freedom parameters.

Usage

```
stmar_to_gstmar(
  gsmar,
  maxdf = 100,
  estimate,
```

```
  calc_std_errors,
  maxit = 100,
  custom_h = NULL
)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.
maxdf	regimes with degrees of freedom parameter value larger than this will be turned into GMAR type.
estimate	set `TRUE` if the new model should be estimated with a variable metric algorithm using the StMAR model parameter value as the initial value. By default `TRUE` iff the model contains data.
calc_std_errors	set `TRUE` if the approximate standard errors should be calculated. By default `TRUE` iff the model contains data.
maxit	the maximum number of iterations for the variable metric algorithm. Ignored if `estimate==FALSE`.
custom_h	A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If `NULL` (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is `6e-6` for the other parameters.

Details

If a StMAR model contains large estimates for the degrees of freedom parameters, one should consider switching to the corresponding G-StMAR model that lets the corresponding regimes to be GMAR type. `stmar_to_gstmar` does this switch conveniently.

Value

Returns an object of class `'gsmar'` defining the specified GMAR, StMAR, or G-StMAR model. If data is supplied, the returned object contains (by default) empirical mixing weights, some conditional and unconditional moments, and quantile residuals. Note that the first p observations are taken as the initial values so the mixing weights, conditional moments, and quantile residuals start from the p+1:th observation (interpreted as t=1).

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

fitGSMAR, GSMAR, iterate_more, get_gradient, get_regime_means, swap_parametrization, stmar_to_gstmar

Examples

```
# These are long running example that take approximately 15 seconds to run.
fit42t <- fitGSMAR(data=M10Y1Y, p=4, M=2, model="StMAR", ncalls=1, seeds=6)
fit42t # Overly large degrees of freedom estimate!

# Switch to the appropriate G-StMAR model:
fit42gs <- stmar_to_gstmar(fit42t)
fit42gs
```

---

swap_parametrization *Swap the parametrization of object of class 'gsmar' defining a GMAR, StMAR, or G-StMAR model*

Description

swap_parametrization swaps the parametrization of object of class 'gsmar' to "mean" if the current parametrization is "intercept", and vice versa.

Usage

```
swap_parametrization(gsmar, calc_std_errors = TRUE, custom_h = NULL)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.
calc_std_errors	should approximate standard errors be calculated?
custom_h	A numeric vector with same the length as the parameter vector: i:th element of custom_h is the difference used in central difference approximation for partial differentials of the log-likelihood function for the i:th parameter. If NULL (default), then the difference used for differentiating overly large degrees of freedom parameters is adjusted to avoid numerical problems, and the difference is 6e-6 for the other parameters.

Details

swap_parametrization is a convenient tool if you have estimated the model in "intercept"-parametrization but wish to work with "mean"-parametrization in the future, or vice versa. For example, approximate standard errors are readily available for parametrized parameters only.

Value

Returns an object of class 'gsmar' defining the specified GMAR, StMAR, or G-StMAR model. If data is supplied, the returned object contains (by default) empirical mixing weights, some conditional and unconditional moments, and quantile residuals. Note that the first p observations are taken as the initial values so the mixing weights, conditional moments, and quantile residuals start from the p+1:th observation (interpreted as t=1).

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

fitGSMAR, GSMAR, iterate_more, get_gradient, get_regime_means, swap_parametrization, stmar_to_gstmar

Examples

```
# G-StMAR model with intercept parametrization
params42gs <- c(0.04, 1.34, -0.59, 0.54, -0.36, 0.01, 0.06, 1.28, -0.36,
                0.2, -0.15, 0.04, 0.19, 9.75)
gstmar42 <- GSMAR(data=M10Y1Y, p=4, M=c(1, 1), params=params42gs,
                  model="G-StMAR")
summary(gstmar42)

# Swap to mean parametrization
gstmar42 <- swap_parametrization(gstmar42)
summary(gstmar42)
```

T10Y1Y *Spread between 10-Year and 1-Year Treasury rates: T10Y1Y*

Description

A dataset containing monthly U.S. interest rate spread between the 10-Year Treasury constant maturity rate and 1-Year Treasury constant maturity rate from 1953IV to 2020II.

Usage

```
T10Y1Y
```

Format

A class 'ts' time series object containing 803 observations.

Source

https://fred.stlouisfed.org/series/GS10 https://fred.stlouisfed.org/series/GS1

TBFF *Spread between the 3-month Treasury bill rate and the effective federal funds rate: TBFF*

Description

A dataset containing the monthly U.S. interest rate spread between the 3-month Treasury bill secondary market rate and the effective federal funds rate from 1954 July to 2019 July (781 observations). This series was studied in the empirical application of Virolainen (2021) introducing the G-StMAR model.

Usage

```
TBFF
```

Format

A class 'ts' time series object containing 781 observations.

Source

https://fred.stlouisfed.org/series/TB3SMFFM

References

- Virolainen S. 2021. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, doi: 10.1515/snde-2020-0060

uncond_moments *Calculate unconditional mean, variance, first p autocovariances and autocorrelations of the GSMAR process.*

Description

uncond_moments calculates the unconditional mean, variance, and the first p autocovariances and autocorrelations of the GSMAR process.

Usage

```
uncond_moments(gsmar)
```

Arguments

gsmar	a class 'gsmar' object, typically generated by fitGSMAR or GSMAR.

Value

Returns a list containing the unconditional mean, variance, and the first p autocovariances and autocorrelations. Note that the lag-zero autocovariance/correlation is not included in the "first p" but is given in the uncond_variance component separately.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: cond_moments(), get_regime_autocovs(), get_regime_means(), get_regime_vars()

Examples

```
# GMAR model
params13 <- c(1.4, 0.88, 0.26, 2.46, 0.82, 0.74, 5.0, 0.68, 5.2, 0.72, 0.2)
gmar13 <- GSMAR(p=1, M=3, params=params13, model="GMAR")
uncond_moments(gmar13)

# StMAR model
params12t <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 100, 3.6)
stmar12t <- GSMAR(p=1, M=2, params=params12t, model="StMAR")
uncond_moments(stmar12t)
```

```
# G-StMAR model (similar to the StMAR model above)
params12gs <- c(1.38, 0.88, 0.27, 3.8, 0.74, 3.15, 0.8, 3.6)
gstmar12 <- GSMAR(p=1, M=c(1, 1), params=params12gs, model="G-StMAR")
uncond_moments(gstmar12)
```

`Wald_test` *Perform Wald test*

Description

`Wald_test` performs a Wald test for a GMAR, StMAR, or G-StMAR model.

Usage

```
Wald_test(gsmar, A, c, h = 6e-06)
```

Arguments

`gsmar`	a class 'gsmar' object, typically generated by `fitGSMAR` or `GSMAR`.
`A`	a size $(k \times n_params)$ matrix with full row rank specifying a part of the null hypothesis, where n_params is the number of parameters in the (unconstrained) model. See details for more information.
`c`	a length k vector specifying a part of the null hypothesis. See details for more information.
`h`	the difference used to approximate the derivatives.

Details

Denoting the true parameter value by θ_0, we test the null hypothesis $A\theta_0 = c$. Under the null, the test statistic is asymptotically χ^2-distributed with k (`=nrow(A)`) degrees of freedom. The parameter θ_0 is assumed to have the same form as in the model supplied in the argument `gsmar` and it is presented in the documentation of the argument `params` in the function `GSMAR` (see `?GSMAR`).

Note that this function does **not** check whether the specified constraints are feasible (e.g., whether the implied constrained model would be stationary or have positive definite error term covariance matrices).

Value

A list with class "htest" containing the following components:

`statistic`	the value of the Wald statistics.
`parameter`	the degrees of freedom of the Wald statistic.
`p.value`	the p-value of the test.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating the type of the test (Wald test).

data.name	a character string giving the names of the supplied model, constraint matrix A, and vector c.
gsmar	the supplied argument gsmar.
A	the supplied argument A.
c	the supplied argument c.
h	the supplied argument h.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**(2), 247-266.
- Meitz M., Preve D., Saikkonen P. 2023. A mixture autoregressive model based on Student's t-distribution. *Communications in Statistics - Theory and Methods*, **52**(2), 499-515.
- Virolainen S. 2022. A mixture autoregressive model based on Gaussian and Student's t-distributions. Studies in Nonlinear Dynamics & Econometrics, **26**(4) 559-580.

See Also

LR_test, fitGSMAR, GSMAR, diagnostic_plot, profile_logliks, quantile_residual_tests, cond_moment_plot

Examples

```
# GMAR p=1, M=2 model:
fit12 <- fitGSMAR(simudata, p=1, M=2, model="GMAR", ncalls=1, seeds=1)

# Test with Wald test whether the AR coefficients are the same in both
# regimes:
# There are 7 parameters in the model and the AR coefficient of the
# first regime is the 2nd element, whereas the AR coefficient of the second
# regime is in the 5th element.
A <- matrix(c(0, 1, 0, 0, -1, 0, 0), nrow=1, ncol=7)
c <- 0
Wald_test(fit12, A=A, c=c)
```

Index