Package 'uGMAR'

April 4, 2020

Title Estimate Univariate Gaussian or Student's t Mixture Autoregressive Model

Version 3.2.5

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 (2018) <arXiv:1805.04010>,
Savi Virolainen (2020) <arXiv:2003.05221>.

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

LazyData true

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

Suggests gsl (>= 1.9-10.3), testthat, knitr, rmarkdown

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Author Savi Virolainen [aut, cre]

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

Repository CRAN

Date/Publication 2020-04-04 12:50:02 UTC

R topics documented:

`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	object of class 'gsmar' created with the function 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# Restricted G-StMAR-model without data
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
gstmar42r <- GSMAR(p=4, M=c(1, 1), params=params42gsr,
 model="G-StMAR", restricted=TRUE)
gstmar42r

# Add data to the model
```

```
gstmar42r <- add_data(data=T10Y1Y, gstmar42r)
gstmar42r
```

`add_dfs` *Add random dfs to a vector*

Description

`add_dfs` adds random degrees of freedom parameters to a vector.

Usage

```
add_dfs(x, how_many)
```

Arguments

`x`	a vector to add the dfs to
`how_many`	how many dfs?

Details

Read the source code for details.

Value

Returns `c(x,dfs)` with `how_many` dfs-elements.

`all_pos_ints` *Check whether all arguments are stricly positive natural numbers*

Description

`all_pos_ints` tells whether all the elements in a vector are strictly positive natural numbers.

Usage

```
all_pos_ints(x)
```

Arguments

`x`	a vector containing the elements to be tested.

Value

Returns `TRUE` or `FALSE` accordingly.

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	object of class 'gsmar' created with the function 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 estination 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 used, 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`.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# These are long running examples and use parallel computing
fit43t <- fitGSMAR(T10Y1Y, 4, 3, model="StMAR", ncalls=2, seeds=1:2)
fit43t
fit43t2 <- alt_gsmar(fit43t, which_largest=2)
fit43t2
```

`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	object of class 'gsmar' created with the function 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)

# StMAR model:
params43 <- c(0.09, 1.31, -0.46, 0.33, -0.23, 0.04, 0.01, 1.15,
 -0.3, -0.03, 0.03, 1.54, 0.06, 1.19, -0.3, 0.42, -0.4, 0.01,
  0.57, 0.22, 8.05, 2.02, 1000)
stmar43 <- GSMAR(T10Y1Y, 4, 3, params43, model="StMAR")
get_gradient(stmar43)
get_foc(stmar43)
get_hessian(stmar43)
get_soc(stmar43)
```

`changeRegime` *Change the specified regime of parameter vector to the given regime-parameter vector*

Description

`changeRegime` changes the specified regime of the parameter vector to correspond the given regime-parameter vector and returns the modified parameter vector. Does not affect mixing weight parameters.

Usage

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

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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

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

`regimeParams` a numeric vector specifying the parameter values that should be inserted to the specified regime.

For non-restricted models: For GMAR model: Size $(p+2x1)$ vector $(\phi_{m,0}, \phi_{m,1}, ..., \phi_{m,p}, \sigma_m^2)$.
For StMAR model: Size $(p+3x1)$ vector $(\phi_{m,0}, \phi_{m,1}, ..., \phi_{m,p}, \sigma_m^2, \nu_m)$.
For G-StMAR model: Same as GMAR for GMAR-components and same as StMAR for StMAR-components.
With linear constraints: Parameter vector as described above, but vector ϕ_m replaced with vector ψ_m that satisfies $\phi_m = R_m\psi_m$.
For restricted models: For GMAR model: Size $(2x1)$ vector $(\phi_{m,0}, \sigma_m^2)$.
For StMAR model: Size $(3x1)$ vector $(\phi_{m,0}, \sigma_m^2, \nu_m)$.
For G-StMAR model: Same as GMAR for GMAR-components and same as StMAR for StMAR-components.
With linear constraints: Parameter vector as described above.

`regime` a positive integer in the interval [1, M] defining which regime should be changed.

Value

Returns modified parameter vector of the form described in `params`.

`change_parametrization`

Change parametrization of a parameter vector

Description

`change_parametrization` changes the parametrization of the given parameter vector to `change_to`.

Usage

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

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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

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

`change_to` either "intercept" or "mean" specifying to which parametrization it should be switched to. If set to `"intercept"`, it's assumed that `params` is mean-parametrized, and if set to `"mean"` it's assumed that `params` is intercept-parametrized.

Value

Returns parameter vector described in `params` but with parametrization changed from intercept to mean (when `change_to==mean`) or from mean to intercept (when `change_to==intercept`).

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

---

checkAndCorrectData *Check that the data is set correctly and correct if not*

Description

`checkAndCorrectData` checks that the data is set correctly and throws an error if there is something wrong with the data.

Usage

```
checkAndCorrectData(data, p)
```

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.

Value

Returns the data as a class 'ts' object.

`checkConstraintMat` *Check the constraint matrices*

Description

`checkConstraintMat` checks for some parts that the constraint matrices are correctly set.

Usage

```
checkConstraintMat(p, M, 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 (2x1) 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`.
`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 applied to the autoregressive parameters. **For non-restricted models:** a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$. Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is **not** desired.

Value

Doesn't return anything but throws an informative error if finds out that something is wrong.

checkPM *Check that p and M are correctly set*

Description

checkPM checks that the arguments p and M are correctly set.

Usage

```
checkPM(p, M, model = c("GMAR", "StMAR", "G-StMAR"))
```

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 (2x1) 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*.

Value

Doesn't return anything but throws an informative error if something is wrong.

check_data *Check that given object contains data*

Description

check_data checks that a given object contains data.

Usage

```
check_data(object)
```

Arguments

object	an object to be tested

Value

Doesn't return anything but throws and error if something is wrong.

check_gsmar *Check that given object has class attribute 'gsmar'*

Description

check_gsmar checks that the given object has class attribute 'gsmar'.

Usage

```
check_gsmar(object)
```

Arguments

object	an object to be tested

Value

Doesn't return anything but throws and error if something is wrong.

check_model *Check that the argument 'model' is correctly specified.*

Description

check_model checks that the argument 'model' is correctly specified.

Usage

```
check_model(model)
```

Arguments

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*.

Value

Doesn't return anything but throws and error if something is wrong.

check_params_length *Check that the parameter vector has the correct dimension*

Description

`check_model` checks that the parameter vector has the correct dimension.

Usage

```
check_params_length(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, . **With linear constraints:** Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

Value

Doesn't return anything but throws and error if something is wrong.

---

condmomentPlot *Condinional mean or variance plot for GMAR, StMAR, and G-StMAR models*

Description

`condmomentPlot` plots the 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 regimiwise conditional means/variances multiplied with mixing weights - summing to the total conditional mean/veriance.

Usage

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

Arguments

`gsmar` object of class `'gsmar'` created with the function `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

`condmomentPlot` 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

`profile_logliks`, `diagnosticPlot`, `fitGSMAR`, `GSMAR`, `quantileResidualTests`, `quantileResidualPlot`

Examples

```
# GMAR model
fit12 <- fitGSMAR(simudata, p=1, M=2, model="GMAR")
condmomentPlot(fit12, which_moment="mean")
condmomentPlot(fit12, which_moment="variance")

# Restricted StMAR model: plot also the individual statistics with
# their approximate critical bounds using the given data
fit42r <- fitGSMAR(T10Y1Y, p=4, M=2, model="StMAR", restricted=TRUE)
condmomentPlot(fit42r, which_moment="mean")
condmomentPlot(fit42r, which_moment="variance")

# G-StMAR model with one GMAR type and one StMAR type regime
fit42g <- fitGSMAR(T10Y1Y, p=4, M=c(1, 1), model="G-StMAR")
condmomentPlot(fit42g, which_moment="mean")
condmomentPlot(fit42g, which_moment="variance")
```

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

Description

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

Usage

```
condMoments(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M).

For StMAR model: Size $(4M+p-1x1)$ vector $(\theta,\nu)=(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,...,\alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector $(\theta,\nu)=(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,.$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies $\phi=R\psi$, where $\psi=(\psi_1,...,\psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying $\phi=C\psi$, where $\phi=(\phi_1,...,\phi_p)$ and $\psi=\psi_1,...,\psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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)xM) matrix containing the regime specific conditional means.

if `to_return=="regime_cvars"`**:** a size ((n_obs-p)xM) matrix containing the regime specific conditional variances.

if `to_return=="total_cmeans"`**:** a size ((n_obs-p)x1) vector containing the total conditional means.

if `to_return=="total_cvars"`**:** a size ((n_obs-p)x1) 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# GMAR model
params12 <- c(0.01, 0.99, 0.02, 0.03, 0.91, 0.32, 0.86)
rcm12 <- condMoments(T10Y1Y, 1, 2, params12, to_return="regime_cmeans")
rcv12 <- condMoments(T10Y1Y, 1, 2, params12, to_return="regime_cvars")
tcm12 <- condMoments(T10Y1Y, 1, 2, params12, to_return="total_cmeans")
tcv12 <- condMoments(T10Y1Y, 1, 2, params12, to_return="total_cvars")

# StMAR model
params43 <- c(0.09, 1.31, -0.46, 0.33, -0.23, 0.04, 0.01, 1.15,
 -0.3, -0.03, 0.03, 1.54, 0.06, 1.19, -0.3, 0.42, -0.4, 0.01,
  0.57, 0.22, 8.05, 2.02, 10000)
rcm43t <- condMoments(T10Y1Y, 4, 3, params43, model="StMAR",
 to_return="regime_cmeans")
rcv43t <- condMoments(T10Y1Y, 4, 3, params43, model="StMAR",
 to_return="regime_cvars")

# G-StMAR model
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
rcv42gsr <- condMoments(T10Y1Y, 4, c(1, 1), params42gsr, model="G-StMAR",
 restricted=TRUE, to_return="regime_cvars")
tcv42gs <- condMoments(T10Y1Y, 4, c(1, 1), params42gsr, model="G-StMAR",
 restricted=TRUE, to_return="total_cvars")
```

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

Description

diagnosticPlot 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

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

Arguments

gsmar	object of class 'gsmar' created with the function 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

diagnosticPlot only plots to a graphical device and does not return anything. Use the function quantileResidualTests 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

profile_logliks, get_foc, fitGSMAR, condmomentPlot, quantileResidualTests, quantileResidualPlot, simulateGSMAR

Examples

```
# StMAR model
fit42 <- fitGSMAR(data=T10Y1Y, p=4, M=2, model="StMAR")
diagnosticPlot(fit42)

# Restricted StMAR model: plot also the individual statistics with
# their approximate critical bounds using the given data
fit42r <- fitGSMAR(T10Y1Y, 4, 2, model="StMAR", restricted=TRUE)
diagnosticPlot(fit42r, nlags=10, nsimu=1, plot_indstats=TRUE)

# Non-mixture version of StMAR model
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
diagnosticPlot(fit101t)

# G-StMAR model
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
diagnosticPlot(fit42g)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(T10Y1Y, 2, 2, constraints=constraints)
diagnosticPlot(fit22c)

# 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.
fit32rc <- fitGSMAR(T10Y1Y, 3, 2, model="StMAR", restricted=TRUE,
 constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
diagnosticPlot(fit32rc)
```

`extractRegime` *Extract regime from a parameter vector*

Description

`extractRegime` extracts the specified regime from the GMAR, StMAR, or G-StMAR model parameter vector. Doesn't extract mixing weight parameter α.

Usage

```
extractRegime(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  regime,
  with_dfs = TRUE
)
```

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

`regime` a positive integer in the interval [1, M] defining which regime should be extracted.

`with_dfs` Should the degrees of freedom parameter (if any) be included?

Value

Returns a numeric vector corresponding to the regime with...

For non-restricted models: For GMAR model: Size $(p+2x1)$ vector $(\phi_{m,0}, \phi_{m,1}, ..., \phi_{m,p}, \sigma_m^2)$.

For StMAR model: Size $(p + 3x1)$ vector $(\phi_{m,0}, \phi_{m,1}, ..., \phi_{m,p}, \sigma_m^2, \nu_m)$.

For G-StMAR model: Same as GMAR for GMAR-components and same as StMAR for StMAR-components.

With linear constraints: Parameter vector as described above, but vector ϕ_m replaced with vector ψ_m that satisfies ϕ_m=$R_m\psi_m$.

For restricted models: For GMAR model: Size $(2x1)$ vector $(\phi_{m,0}, \sigma_m^2)$.

For StMAR model: Size $(3x1)$ vector $(\phi_{m,0}, \sigma_m^2, \nu_m)$.

For G-StMAR model: Same as GMAR for GMAR-components and same as StMAR for StMAR-components.

With linear constraints: Parameter vector as described above.

`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 = min(2, ncalls, parallel::detectCores()),
  maxit = 300,
  seeds = NULL,
  printRes = TRUE,
  runTests = 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 (2x1) 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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.

`printRes` should the estimation results be printed?

`runTests` should quantile residuals tests be performed after the estimation?

`...` 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 ('redudant 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`.

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 parameters 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.

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, quantile residuals, and quantile residual test results if the tests were performed. 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-likelihood values from all of the estimation rounds. The estimated parameter vector can be obtained as `gsmar$params` (and the corresponding approximate standard errors as `gsmar$std_errors`) and it's...

For non-restricted models: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where $\upsilon_m = (\phi_{m,0}, \boldsymbol{\phi}_m, \sigma_m^2)$ and $\boldsymbol{\phi}_m$=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$ and provide a list of constraint matrices **C** that satisfy $\boldsymbol{\phi}_m = C_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\psi}_m = (\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: For GMAR model: Size $(3M+p-1x1)$ vector $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where $\boldsymbol{\phi}$=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector $(\theta, \nu) = (\phi_{1,0}, ..., \phi_{M,0}, \boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M)$.

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector $(\theta, \nu) = (\phi_{1,0}, ..., \phi_{M,0}, \boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, .$

With linear constraints: Replace the vector $\boldsymbol{\phi}$ with vector $\boldsymbol{\psi}$ and provide a constraint matrix C that satisfies $\boldsymbol{\phi} = C\boldsymbol{\psi}$, where $\boldsymbol{\psi} = (\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model the degrees of freedom parameters ν_m have to be larger than 2.

S3 methods

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

Suggested packages

For faster evaluation of the quantile residuals for StMAR and G-StMAR models, install the suggested package "gsl". Note that for large StMAR and G-StMAR models with large data, performing the quantile residual tests may take significantly long time without the package "gsl".

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- 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. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

GSMAR, iterate_more, , stmar_to_gstmar, add_data, profile_logliks, swap_parametrization, get_gradient, simulateGSMAR, predict.gsmar, diagnosticPlot, quantileResidualTests, condMoments, uncondMoments

Examples

```
# These are long running examples that use parallel computing

# GMAR model
fit12 <- fitGSMAR(simudata, p=1, M=2, model="GMAR")
summary(fit12)
plot(fit12)

# StMAR model
fit42 <- fitGSMAR(data=T10Y1Y, p=4, M=2, model="StMAR")
fit42
summary(fit42)
plot(fit42)

# Restricted StMAR model: plot also the individual statistics with
```

```
# their approximate critical bounds using the given data
fit42r <- fitGSMAR(T10Y1Y, 4, 2, model="StMAR", restricted=TRUE)
fit42r
plot(fit42)

# Non-mixture version of StMAR model
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
diagnosticPlot(fit101t)

# G-StMAR model with one GMAR type and one StMAR type regime
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
diagnosticPlot(fit42g)

# GMAR model; seeds for rerpoducibility
fit43gm <- fitGSMAR(T10Y1Y, 4, M=3, model="GMAR", ncalls=16,
  seeds=1:16)
fit43gm

# Restricted GMAR model
fit43gmr <- fitGSMAR(T10Y1Y, 4, M=3, model="GMAR", ncalls=12,
  restricted=TRUE, seeds=1:12)
fit43gmr


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

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(T10Y1Y, 2, 2, constraints=constraints)
fit22c

# Such constrained StMAR(3, 1) model that the second order AR coefficient
# is constrained to zero.
constraints <- list(matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
fit31tc <- fitGSMAR(T10Y1Y, 3, 1, model="StMAR", constraints=constraints)
fit31tc

# 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.
fit32rc <- fitGSMAR(T10Y1Y, 3, 2, model="StMAR", restricted=TRUE,
 constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
fit32rc
```

`format_valuef` *Function factory for formatting values*

Description

format_valuef generates functions that format values so that they print out with the desired number of digits.

Usage

```
format_valuef(digits)
```

Arguments

digits	number of digits to use

Value

Returns a function that takes an atomic vector as its argument and returns it formatted to character with digits decimals.

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,
  smartMu = min(100, ceiling(0.5 * ngen)),
  meanscale,
  sigmascale,
  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 (2x1) 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 applied to the autoregressive parameters. **For non-restricted models:** a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$. Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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.
`smartMu`	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.
`meanscale`	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.

`sigmascale` 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$C_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$C\psi$, where ψ= (ψ_1, ..., ψ_q).

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and υ a degrees of freedom parameter. Note that in the case **M=1**, the parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m have to be larger than 2. If not specified (or `FALSE` as is default), the initial population will be drawn randomly.

`regime_force_scale`

a non-negative real number specifying how much should natural selection favour 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 favouring or large number for heavy favouring. Without any favouring the genetic algorithm gets more often stuck in an area of the parameter space where some regimes are wasted, but with too much favouring 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(mixingWeights[,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`.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- 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. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

getOmega *Generate the covariance matrix Omega for quantile residual tests*

Description

getOmega generates the covariance matrix Omega used in the quantile residual tests.

Usage

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

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`g` a function specifying the transformation.

`dim_g` output dimension of the transformation `g`.

Details

This function is used for quantile residuals tests in `quantileResidualTests`.

Value

Returns size (`dim_g`x`dim_g`) covariance matrix Omega.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

`quantileResidualTests`

`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`	object of class `'gsmar'` created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

Examples

```
params12 <- c(1.7, 0.85, 0.3, 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_IC *Calculate AIC, HQIC and BIC*

Description

`get_IC` calculates AIC, HQIC and BIC

Usage

```
get_IC(loglik, npars, obs)
```

Arguments

`loglik`	log-likelihood value
`npars`	the number of (freely estimated) parameters in the model.
`obs`	the number of observations with initial values excluded for conditional models.

Value

Returns a data frame containing the information criteria values.

get_minval *Returns the default smallest allowed log-likelihood for given data.*

Description

`get_minval` returns the default smallest allowed log-likelihood for given data.

Usage

```
get_minval(data)
```

Arguments

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

Details

This function exists simply to avoid dublication inside the package.

Value

Returns `-(10^(ceiling(log10(length(data))) + 1) -1)`

See Also

`fitGSMAR`, `GAfit`

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	object of class 'gsmar' created with the function 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- There are currently no published references for the G-StMAR model, but it's a straightforward generalization with theoretical properties similar to the GMAR and StMAR models.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: condMoments(), get_regime_means(), get_regime_vars(), uncondMoments()

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`	object of class `'gsmar'` created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

`condMoments`, `uncondMoments`, `get_regime_vars`, `get_regime_autocovs`

Other moment functions: `condMoments()`, `get_regime_autocovs()`, `get_regime_vars()`, `uncondMoments()`

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`	object of class `'gsmar'` created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- There are currently no published references for the G-StMAR model, but it's a straightforward generalization with theoretical properties similar to the GMAR and StMAR models.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: `condMoments()`, `get_regime_autocovs()`, `get_regime_means()`, `uncondMoments()`

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)
```

---

get_varying_h *Get differences 'h' which are adjusted for overly large degrees of freedom parameters*

Description

get_varying_h adjusts differences for overly large degrees of freedom parameters for finite difference approximation of the derivatives of the log-likelihood function. StMAR and G-StMAR models are supported.

Usage

```
get_varying_h(p, M, params, model)
```

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}$,$\phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where ϕ=$(\phi_1, ..., \phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}$,$\phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}$,$\phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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*.

Details

This function is used for approximating gradient and Hessian of a StMAR or G-StMAR model. Large degrees of freedom parameters cause significant numerical error if too small differences are used.

Value

Returns a vector with the same length as `params`. For other parameters than degrees of freedom parameters larger than 100, the differences will be `6e-6`. For the large degrees of freedom parameters, the difference will be `signif(df/1000,digits=2)`.

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where ϕ=$(\phi_1, ..., \phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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 polynomias for each regime, and several unconditional moments. Supported for models that include data only.

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 Studen'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 dotter 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).

Methods (by generic)

- logLik: Log-likelihood method
- residuals: residuals method to extract quantile residuals
- summary: summary method, standard errors in brackets
- plot: Plot method for class 'gsmar'
- print: 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

fitGSMAR, iterate_more, add_data, stmar_to_gstmar, swap_parametrization, get_gradient, simulateGSMAR, predict.gsmar, condMoments, uncondMoments

Examples

```
# GMAR model without data
params12 <- c(0.18, 0.93, 0.01, 0.86, 0.68, 0.02, 0.88)
gmar12 <- GSMAR(p=1, M=2, params=params12, model="GMAR")
gmar12

# 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

# Restricted G-StMAR-model
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
gstmar42r <- GSMAR(data=T10Y1Y, p=4, M=c(1, 1), params=params42gsr,
 model="G-StMAR", restricted=TRUE)
gstmar42r

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- c(0.03, 1.27, -0.29, 0.03, -0.01, 0.91, 0.34, 0.88)
gmar22c <- GSMAR(T10Y1Y, p=2, M=2, params=params22c,
 model="GMAR", constraints=constraints)
gmar22c
```

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

Description

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

Usage

```
isStationary(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, . **With linear constraints:** Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q). 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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

Examples

```
# GMAR model
params22 <- c(0.4, 0.39, 0.6, 0.3, 0.4, 0.1, 0.6, 0.3, 0.8)
isStationary(2, 2, params22)

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

# G-StMAR model
params12gs <- c(1, 0.1, 1, 2, 0.2, 2, 0.8, 20)
isStationary(1, c(1, 1), 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)
isStationary(1, 3, params13r, restricted=TRUE)

# Restricted StMAR model
```

```
params22tr <- c(-0.1, -0.2, 0.01, 0.99, 0.3, 0.4, 0.9, 3, 13)
isStationary(2, 2, params22tr, model="StMAR", restricted=TRUE)

# Restricted G-StMAR model
params13gsr <- c(1, 2, 3, -0.99, 1, 2, 3, 0.5, 0.4, 20, 30)
isStationary(1, c(1, 2), params13gsr, model="G-StMAR", restricted=TRUE)

# GMAR model as a mixture of AR(2) and AR(1) models
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- c(1.2, 0.8, 0.2, 0.3, 3.3, 0.7, 3, 0.8)
isStationary(2, 2, params22c, constraints=constraints)

# 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)
isStationary(3, 2, params32trc, model="StMAR", restricted=TRUE,
             constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
```

`isStationary_int` *Check the stationarity and identification conditions of specified GMAR, StMAR, or G-StMAR model.*

Description

`isStationary_int` checks the stationarity condition and `isIdentifiable` checks the identification condition of the specified GMAR, StMAR, or G-StMAR model.

Usage

```
isStationary_int(p, M, params, restricted = FALSE)

isIdentifiable(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m have to be larger than 2.

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

`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*.

`constraints` specifies linear constraints applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

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

Details

`isStationary_int` does not support models imposing linear constraints. In order to use it for a model imposing linear constraints, one needs to expand the constraints first to obtain a nonconstrained parameters vector.

Note that `isStationary_int` returns `FALSE` for stationary parameter vectors if they are extremely close to the boundary of the stationarity region.

`isIdentifiable` checks that the regimes are sorted according to the mixing weight parameters and that there are no dublicate regimes.

Value

Returns TRUE or FALSE accordingly.

Warning

These functions don't have any argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2015. Gaussian Mixture Autoregressive model for univariate time series. *Journal of Time Series Analysis*, **36**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

`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`	object of class `'gsmar'` created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# Estimate GMAR model with only 50 generations of genetic algorithm and
# only 1 iteration in variable metric algorithm
fit13 <- fitGSMAR(T10Y1Y, 1, 3, maxit=1, ngen=50, ncalls=1, seeds=1)
fit13

# Iterate more since iteration limit was reached
fit13 <- iterate_more(fit13)
fit13
```

---

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"),
  returnTerms = 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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$, σ_m^2) and $\boldsymbol{\phi}_m$=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M).

With linear constraints: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$ and provide a list of constraint matrices **C** that satisfy $\boldsymbol{\phi}_m$=$R_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\psi}_m$= ($\psi_{m,1}$, ..., ψ_{m,q_m}).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where $\boldsymbol{\phi}$=($\boldsymbol{\phi}_1$, ..., $\boldsymbol{\phi}_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector $\boldsymbol{\phi}$ with vector $\boldsymbol{\psi}$ and provide a constraint matrix C that satisfies $\boldsymbol{\phi}$=$R\boldsymbol{\psi}$, where $\boldsymbol{\psi}$= (ψ_1, ..., ψ_q).

Symbol $\boldsymbol{\phi}$ 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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`returnTerms` 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 `returnTerms`.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

fitGSMAR, GSMAR, quantileResiduals, mixingWeights, calc_gradient

Examples

```
# StMAR model
params43 <- c(0.09, 1.31, -0.46, 0.33, -0.23, 0.04, 0.01, 1.15,
 -0.3, -0.03, 0.03, 1.54, 0.06, 1.19, -0.3, 0.42, -0.4, 0.01,
  0.57, 0.22, 8.05, 2.02, 10000)
loglikelihood(T10Y1Y, p=4, M=3, params=params43, model="StMAR")

# Restricted G-StMAR-model
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
loglikelihood(T10Y1Y, p=4, M=c(1, 1), params=params42gsr, model="G-StMAR",
  restricted=TRUE)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- c(0.03, 1.27, -0.29, 0.03, -0.01, 0.91, 0.34, 0.88)
loglikelihood(T10Y1Y, p=2, M=2, params=params22c, model="GMAR",
 constraints=constraints)
```

---

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

Description

loglikelihood_int computes the log-likelihood of the specified GMAR, StMAR, or G-StMAR model.

Usage

```
loglikelihood_int(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  boundaries = TRUE,
  checks = TRUE,
 to_return = c("loglik", "mw", "mw_tplus1", "loglik_and_mw", "terms", "regime_cmeans",
    "regime_cvars", "total_cmeans", "total_cvars", "qresiduals"),
  minval
)
```

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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

boundaries — a logical argument. If TRUE, then loglikelihood returns minval if...

- some component variance is not larger than zero,
- some parametrized mixing weight $\alpha_1, ..., \alpha_{M-1}$ is not larger than zero,
- sum of the parametrized mixing weights is not smaller than one,
- if the model is not stationary,
- or if model=="StMAR" or model=="G-StMAR" and some degrees of freedom parameter ν_m is not larger than two.

Argument minval will be used only if boundaries==TRUE.

checks — TRUE or FALSE specifying whether argument checks, such as stationarity checks, should be done.

to_return — should the returned object be the log-likelihood value, mixing weights, mixing weights including value for $alpha_{m,T+1}$, a list containing log-likelihood value and mixing weights, the terms $l_t : t = 1, .., T$ in the log-likelihood function (see *KMS 2015, eq.(13)*), regimewise conditional means, regimewise conditional variances, total conditional means, total conditional variances, or quantile residuals? Default is the log-likelihood value ("loglik").

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

Value

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

By default: log-likelihood value of the specified model,

If to_return=="mw": a size ((n_obs-p)xM) matrix containing the mixing weights: for m:th component in the m:th column.

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

If to_return=="loglik_and_mw": a list of two elements. The first element contains the log-likelihood value and the second element contains the mixing weights.

If to_return=="terms": a size ((n_obs-p)x1) numeric vector containing the terms l_t.

If to_return=="regime_cmeans": a size ((n_obs-p)xM) matrix containing the regime specific conditional means.

If to_return=="regime_cvars": a size ((n_obs-p)xM) matrix containing the regime specific conditional variances.

If `to_return=="total_cmeans"`**:** a size ((n_obs-p)x1) vector containing the total conditional means.

If `to_return=="total_cvars"`**:** a size ((n_obs-p)x1) vector containing the total conditional variances.

If `to_return=="qresiduals"`**:** a size ((n_obs-p)x1) vector containing the quantile residuals.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

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

Description

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

Usage

```
mixingWeights(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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)xM) matrix containing the mixing weights: for m:th component in m:th column.

If `to_return=="mw_tplus1"`**:** a size ((n_obs-p+1)xM) 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

Examples

```
# StMAR model
params43 <- c(0.09, 1.31, -0.46, 0.33, -0.23, 0.04, 0.01, 1.15,
 -0.3, -0.03, 0.03, 1.54, 0.06, 1.19, -0.3, 0.42, -0.4, 0.01,
  0.57, 0.22, 8.05, 2.02, 10000)
mixingWeights(T10Y1Y, p=4, M=3, params=params43, model="StMAR")

# Restricted G-StMAR-model
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
mixingWeights(T10Y1Y, p=4, M=c(1, 1), params=params42gsr, model="G-StMAR",
  restricted=TRUE)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- c(0.03, 1.27, -0.29, 0.03, -0.01, 0.91, 0.34, 0.88)
mixingWeights(T10Y1Y, p=2, M=2, params=params22c, model="GMAR",
 constraints=constraints)
```

`mixingWeights_int` *Calculate mixing weights of a GMAR, StMAR, or G-StMAR model*

Description

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

Usage

```
mixingWeights_int(
  data,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  parametrization = c("intercept", "mean"),
  checks = TRUE,
  to_return = c("mw", "mw_tplus1")
)
```

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where ϕ=$(\phi_1, ..., \phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`checks` `TRUE` or `FALSE` specifying whether argument checks, such as stationarity checks, should be done.

`to_return` should the returned object contain mixing weights for t=1,..,T (`"mw"`) or for t=1,..,T+1 (`"mw_tplus1"`)?

Details

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

Value

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

If `to_return=="mw_tplus1"`**:** a size ((n_obs-p+1)xM) 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

`nParams` *Calculate the number of parameters*

Description

`nParams` calculates the number of parameters that should be in the parameter vector.

Usage

```
nParams(
  p,
  M,
  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 (2x1) 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

Value

Returns the number of parameters.

`parameterChecks` *Check the parameter vector is specified correctly*

Description

`parameterChecks` checks dimension, restrictions, and stationarity of the given parameter of a GMAR, StMAR, or G-StMAR model. Throws an error if any check fails. Does NOT consider the identifiability condition!

Usage

```
parameterChecks(
  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 (2x1) 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 GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= $(\phi_{m,0}$,$\phi_m, \sigma_m^2)$ and ϕ_m=$(\phi_{m,1}, ..., \phi_{m,p}), m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ,ν)=(υ_1,...,υ_M, $\alpha_1,...,\alpha_{M-1},\nu_{M1+1},...,\nu_M$).

`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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1,...,\phi_p)$ and ψ= $\psi_1,...,\psi_q$.

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

Value

Throws an informative error if any check fails. Does not return anything.

`pick_alphas` *Pick mixing weights parameters from parameter vector*

Description

`pick_alphas` picks and returns the mixing weights parameters (including the non-parametrized one for the last component) from the given parameter vector.

Usage

```
pick_alphas(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m, \sigma_m^2$) and $\boldsymbol{\phi}_m$=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$ and provide a list of constraint matrices **C** that satisfy $\boldsymbol{\phi}_m$=$R_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\psi}_m$= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where $\boldsymbol{\phi}$=($\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}, \sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

With linear constraints: Replace the vector $\boldsymbol{\phi}$ with vector $\boldsymbol{\psi}$ and provide a constraint matrix C that satisfies $\boldsymbol{\phi}$=$R\boldsymbol{\psi}$, where $\boldsymbol{\psi}$= ($\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_q$).

Symbol $\boldsymbol{\phi}$ 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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_m) constraint matrices C_m of full column rank satisfying $\boldsymbol{\phi}_m$=$C_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\phi}_m$= ($\boldsymbol{\phi}_{m,1}, ..., \boldsymbol{\phi}_{m,p}$) and $\boldsymbol{\psi}_m$= ($\boldsymbol{\psi}_{m,1}, ..., \boldsymbol{\psi}_{m,q_m}$).

For restricted models: a size (pxq) constraint matrix C of full column rank satisfying $\boldsymbol{\phi}$=$C\boldsymbol{\psi}$, where $\boldsymbol{\phi}$= ($\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_p$) and $\boldsymbol{\psi}$= $\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_q$.

Symbol $\boldsymbol{\phi}$ denotes an AR coefficient. Note that regardless of any constraints, the nominal autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is **not** desired.

Value

Returns a vector of length `M` containing the mixing weight parameters α_m.

pick_dfs *Pick degrees of freedom parameters from a parameter vector*

Description

pick_dfs picks and returns the degrees of freedom parameters from the given parameter vector.

Usage

```
pick_dfs(p, M, params, model = c("GMAR", "StMAR", "G-StMAR"))
```

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, . **With linear constraints:** Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q). 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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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*.

Value

Returns a vector of length `M` or `M2` containing the degrees of freedom parameters.

`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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

Value

Returns a $(Mx(p + 2))$ 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.

`pick_phi0` *Pick phi0 or mean parameters from parameter vector*

Description

`pick_phi0` picks and returns the phi0 or mean parameters from parameter vector.

Usage

```
pick_phi0(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

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

Value

Returns a vector of length `M` containing the phi0 or mean parameters depending parametrization.

`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.

Examples

```
# GMAR model
params12 <- c(1.7, 0.85, 0.3, 4.12, 0.73, 1.98, 0.63)
gmar12 <- GSMAR(simudata, 1, 2, params12)
pred <- predict(gmar12, n_ahead=10, plotRes=FALSE, pi=c(0.8, 0.9, 0.99), pi_type="two-sided")
plot(pred, nt=50)
```

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

Description

quantileResidualTests 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)

quantileResidualTests(
  gsmar,
  lagsAC = c(1, 2, 5, 10),
  lagsCH = lagsAC,
  nsimu = 1,
  printRes = TRUE
)
```

Arguments

x	object of class 'qrtest' created with the function quantileResidualTests.
...	graphical parameters passed to segments in plot.qrtest. Currectly not used in print.qrtest
digits	the number of digits to be print
gsmar	object of class 'gsmar' created with the function fitGSMAR or GSMAR.
lagsAC	a numeric vector of positive integers specifying the lags for which autocorrelation is tested.
lagsCH	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.
printRes	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.

Methods (by generic)

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

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 evaluations may take 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

`profile_logliks`, `fitGSMAR`, `GSMAR`, `diagnosticPlot`, `predict.gsmar`, `getOmega`,

Examples

```
# GMAR model
fit12 <- fitGSMAR(simudata, p=1, M=2, model="GMAR")
qrt <- quantileResidualTests(fit12, lagsAC=c(1, 5, 10, 15))
```

```
# G-StMAR model
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
qrtest42g <- quantileResidualTests(fit42g)
plot(qrtest42g)

# Restricted GMAR model
fit43gmr <- fitGSMAR(T10Y1Y, 4, 3, model="GMAR", restricted=TRUE)
qrtest43gmr <- quantileResidualTests(fit43gmr, lagsAC=1:10)
plot(qrtest43gmr)

# Non-mixture version of StMAR model
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
quantileResidualTests(fit101t, lagsAC=c(1, 2, 5), printRes=FALSE)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(T10Y1Y, 2, 2, constraints=constraints)
quantileResidualTests(fit22c, lagsAC=c(1, 3), printRes=FALSE)
```

`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"),
  plotRes = 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"?
plotRes	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 simulateGSMAR 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

simulateGSMAR, condMoments, fitGSMAR, GSMAR, quantileResidualTests, diagnosticPlot

Examples

```
# StMAR model
fit42 <- fitGSMAR(data=T10Y1Y, p=4, M=2, model="StMAR")
pred42 <- predict(fit42, n_ahead=10, pi=c(0.95, 0.8))
pred42

# Non-mixture StMAR model, upper prediction intervals
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
predict(fit101t, n_ahead=10, pi_type="upper", pi=0.9)

# G-StMAR model, no prediction intervals
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
pred42gs <- predict(fit42g, n_ahead=2, pred_type="median",
 pi_type="none", plotRes=FALSE)
pred42gs
plot(pred42gs)

# Restricted GMAR model, one-step conditional mean prediction
fit43gmr <- fitGSMAR(T10Y1Y, 4, 3, model="GMAR", restricted=TRUE)
pred43gmr <- predict(fit43gmr, pred_type="cond_mean", plotRes=FALSE)
pred43gmr

# 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.
fit32rc <- fitGSMAR(T10Y1Y, 3, 2, model="StMAR", restricted=TRUE,
 constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2))
predict(fit32rc, n_ahead=3, pi_type="lower")
```

---

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

Examples

```
# GMAR-model
params12 <-  c(0.18, 0.93, 0.01, 0.86, 0.68, 0.02, 0.88)
gmar12 <- GSMAR(simudata, 1, 2, params12)
pred <- predict(gmar12, n_ahead=10, plotRes=FALSE)
pred
print(pred, digits=3)
```

---

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

Examples

```
# GMAR model
fit12 <- fitGSMAR(simudata, 1, 2, ncalls=4)
gsmarsum12 <- summary(fit12)
gsmarsum12
print(gsmarsum12, digits=4)
```

`profile_logliks` *Plot profile log-likehoods around the estimates*

Description

`profile_logliks` plots profile log-likelihoods around the estimates.

Usage

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

Arguments

`gsmar`	object of class `'gsmar'` created with the function `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.
`precission`	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 precission 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

quantileResidualPlot, diagnosticPlot, condmomentPlot, GSMAR, quantileResidualTests, simulateGSMAR

Examples

```
# StMAR model
fit42 <- fitGSMAR(data=T10Y1Y, p=4, M=2, model="StMAR")
profile_logliks(fit42)

# Non-mixture version of StMAR model
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
profile_logliks(fit101t)

# G-StMAR model
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
profile_logliks(fit42g)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(T10Y1Y, 2, 2, constraints=constraints)
profile_logliks(fit22c)
```

---

quantileResidualPlot *Plot quantile residual time series and histogram*

Description

quantileResidualsPlot plots quantile residual time series and histogram.

Usage

```
quantileResidualPlot(gsmar)
```

Arguments

gsmar	object of class 'gsmar' created with the function 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

profile_logliks, diagnosticPlot, fitGSMAR, GSMAR, quantileResidualTests, simulateGSMAR

Examples

```
# StMAR model
fit42 <- fitGSMAR(data=T10Y1Y, p=4, M=2, model="StMAR")
quantileResidualPlot(fit42)

# Restricted StMAR model: plot also the individual statistics with
# their approximate critical bounds using the given data
fit42r <- fitGSMAR(T10Y1Y, 4, 2, model="StMAR", restricted=TRUE)
quantileResidualPlot(fit42r)

# Non-mixture version of StMAR model
fit101t <- fitGSMAR(T10Y1Y, 10, 1, model="StMAR", ncores=1, ncalls=1)
quantileResidualPlot(fit101t)

# G-StMAR model
fit42g <- fitGSMAR(T10Y1Y, 4, M=c(1, 1), model="G-StMAR")
quantileResidualPlot(fit42g)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
fit22c <- fitGSMAR(T10Y1Y, 2, 2, constraints=constraints)
quantileResidualPlot(fit22c)
```

---

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

Description

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

Usage

```
quantileResiduals(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, . **With linear constraints:** Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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.

Suggested packages

Install the suggested package "gsl" for faster evaluation of the quantile residuals of StMAR and G-StMAR models.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

Examples

```
# StMAR model
params43 <- c(0.09, 1.31, -0.46, 0.33, -0.23, 0.04, 0.01, 1.15,
 -0.3, -0.03, 0.03, 1.54, 0.06, 1.19, -0.3, 0.42, -0.4, 0.01,
  0.57, 0.22, 8.05, 2.02, 10000)
quantileResiduals(T10Y1Y, p=4, M=3, params=params43, model="StMAR")

# Restricted G-StMAR-model
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
quantileResiduals(T10Y1Y, p=4, M=c(1, 1), params=params42gsr, model="G-StMAR",
  restricted=TRUE)

# Two-regime GMAR p=2 model with the second AR coeffiecient of
# of the second regime contrained to zero.
constraints <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- c(0.03, 1.27, -0.29, 0.03, -0.01, 0.91, 0.34, 0.88)
quantileResiduals(T10Y1Y, p=2, M=2, params=params22c, model="GMAR",
 constraints=constraints)
```

---

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

Description

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

Usage

```
quantileResiduals_int(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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.

Suggested packages

Install the suggested package "gsl" for faster evaluation of the quantile residuals for the StMAR and G-StMAR models.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

---

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

Description

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

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

Usage

```
randomIndividual(
  p,
  M,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  meanscale,
  sigmascale,
  forcestat = FALSE
)

smartIndividual(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  meanscale,
  sigmascale,
  accuracy,
  whichRandom = 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 (2x1) 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 applied to the autoregressive parameters. **For non-restricted models:** a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

`meanscale` 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.

`sigmascale` 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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`.

`whichRandom` a numeric vector of maximum length `M` specifying which regimes should be random instead of "smart" when using `smartIndividual`. 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

```
# GMAR model parameter vector
params22 <- randomIndividual(2, 2, meanscale=c(0, 1), sigmascale=1)
smart22 <- smartIndividual(2, 2, params22, accuracy=10)
cbind(params22, smart22)


# Restricted GMAR parameter vector
params12r <- randomIndividual(1, 2, restricted=TRUE, meanscale=c(-2, 2), sigmascale=2)
smart12r <- smartIndividual(1, 2, params12r, restricted=TRUE, accuracy=20)
cbind(params12r, smart12r)


# StMAR parameter vector: first regime is random in the "smart individual"
params13t <- randomIndividual(1, 3, model="StMAR", meanscale=c(3, 1), sigmascale=3)
smart13t <- smartIndividual(1, 3, params13t, model="StMAR", accuracy=15,
                            meanscale=c(3, 3), sigmascale=3, whichRandom=1)
cbind(params13t, smart13t)


# Restricted StMAR parameter vector
params22tr <- randomIndividual(2, 2, model="StMAR", restricted=TRUE,
                               meanscale=c(3, 2), sigmascale=0.5)
smart22tr <- smartIndividual(2, 2, params22tr, model="StMAR", restricted=TRUE,
                             accuracy=30)
cbind(params22tr, smart22tr)


# G-StMAR parameter vector
params12gs <- randomIndividual(1, c(1, 1), model="G-StMAR", meanscale=c(0, 1),
                               sigmascale=1)
smart12gs <- smartIndividual(1, c(1, 1), params12gs, model="G-StMAR", accuracy=20)
cbind(params12gs, smart12gs)


# Restricted G-StMAR parameter vector
params23gsr <- randomIndividual(2, c(1, 2), model="G-StMAR", restricted=TRUE,
                                meanscale=c(-1, 1), sigmascale=3)
smart23gsr <- smartIndividual(2, c(1, 2), params23gsr, model="G-StMAR", restricted=TRUE,
                              meanscale=c(0, 1), sigmascale=1, accuracy=20, whichRandom=2)
cbind(params23gsr, smart23gsr)
```

```
# GMAR model as a mixture of AR(2) and AR(1) models
C <- list(diag(1, ncol=2, nrow=2), as.matrix(c(1, 0)))
params22c <- randomIndividual(2, 2, constraints=C, meanscale=c(1, 1),
                              sigmascale=1)
smart22c <- smartIndividual(2, 2, params22c, constraints=C, accuracy=10)
cbind(params22c, smart22c)


# Such constrained StMAR(3, 2) model that the second order AR coefficients
# are constrained to zero.
C0 = matrix(c(1, 0, 0, 0, 0, 1), ncol=2)
C = list(C0, C0)
params32c <- randomIndividual(3, 2, model="StMAR", constraints=C,
                              meanscale=c(1, 1), sigmascale=1)
smart32c <- smartIndividual(3, 2, params32c, model="StMAR", constraints=C, accuracy=10)
cbind(params32c, smart32c)


# 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 <- randomIndividual(3, 2, model="StMAR", restricted=TRUE,
                                constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2),
                                meanscale=c(-2, 0.5), sigmascale=4)
smart32trc <- smartIndividual(3, 2, params32trc, model="StMAR", restricted=TRUE,
                              constraints=matrix(c(1, 0, 0, 0, 0, 1), ncol=2),
                              meanscale=c(0, 0.1), sigmascale=0.1, whichRandom=2,
                              accuracy=20)
cbind(params32trc, smart32trc)
```

`randomIndividual_int` *Create random GMAR, StMAR, or G-StMAR model compatible parameter vector*

Description

`randomIndividual_int` creates a random GMAR, StMAR, or G-StMAR model compatible parameter vector.

`smartIndividual_int` creates a random GMAR, StMAR, or G-StMAR model compatible parameter vector close to argument `params`.

Usage

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

```
  meanscale,
  sigmascale,
  forcestat = FALSE
)

smartIndividual_int(
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  meanscale,
  sigmascale,
  accuracy,
  whichRandom,
  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 (2x1) 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 applied to the autoregressive parameters. **For non-restricted models:** a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$. Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal autoregressive order is always p for all regimes. Ignore or set to NULL if applying linear constraints is **not** desired.
meanscale	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.

`sigmascale` 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$, σ_m^2) and $\boldsymbol{\phi}_m$=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M).

With linear constraints: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$ and provide a list of constraint matrices **C** that satisfy $\boldsymbol{\phi}_m$=$R_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\psi}_m$= ($\psi_{m,1}$, ..., ψ_{m,q_m}).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where $\boldsymbol{\phi}$=($\boldsymbol{\phi}_1$, ..., $\boldsymbol{\phi}_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,$\boldsymbol{\phi}$, σ_1^2, ..., σ_M^2, α_1, .

With linear constraints: Replace the vector $\boldsymbol{\phi}$ with vector $\boldsymbol{\psi}$ and provide a constraint matrix C that satisfies $\boldsymbol{\phi}$=$R\boldsymbol{\psi}$, where $\boldsymbol{\psi}$= (ψ_1, ..., ψ_q).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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`.

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

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.

random_arcoefs *Create random AR coefficients*

Description

random_arcoefs generates random AR coefficients.

Usage

```
random_arcoefs(p, forcestat = FALSE, sd = 0.6/p)
```

Arguments

p	a positive integer specifying the autoregressive order of the model.
forcestat	use the algorithm by Monahan (1984) to force stationarity on the AR parameters (slower)?
sd	if forcestat==FALSE, then AR parameters are drawn from zero mean normal distribution with sd given by this parameter.

Details

If forcestat==TRUE, then the AR coefficients are relatively large, otherwise they are usually relatively small.

Value

Returns $px1$ vector containing random AR coefficients.

References

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

random_regime *Create random regime parameters*

Description

random_regime generates random regime parameters.

Usage

```
random_regime(
  p,
  meanscale,
  sigmascale,
  restricted = FALSE,
  constraints = NULL,
  m,
  forcestat = FALSE
)
```

Arguments

p	a positive integer specifying the autoregressive order of the model.
meanscale	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.
sigmascale	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)`.
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 applied to the autoregressive parameters. **For non-restricted models:** a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$. Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal autoregressive order is always `p` for all regimes. Ignore or set to `NULL` if applying linear constraints is **not** desired.
m	which regime? This is required for models with constraints for which a list of possibly differing constraint matrices is provided.
forcestat	use the algorithm by Monahan (1984) to force stationarity on the AR parameters (slower)? Not supported for constrained models.

Details

If `forcestat==TRUE`, then the AR coefficients are relatively large, otherwise they are usually relatively small.

Value

Regular models: $\upsilon_m = (\phi_{m,0}, \boldsymbol{\phi}_m, \sigma_m^2)$ where $\boldsymbol{\phi}_m = (\phi_{m,1}, ..., \phi_{m,p})$.

Restricted models: Not supported!

Constrained models: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$.

References

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

`reformConstrainedPars` *Reform parameter vector with linear constraints to correspond non-constrained parameter vector.*

Description

`reformConstrainedPars` reforms the parameter vector of a model with linear constrains to the "standard form" so that it's comparable with non-constrained models.

Usage

```
reformConstrainedPars(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where $\upsilon_m = (\phi_{m,0}, \boldsymbol{\phi}_m, \sigma_m^2)$ and $\boldsymbol{\phi}_m$=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ,ν)=(υ_1,...,υ_M, $\alpha_1,...,\alpha_{M-1},\nu_{M1+1},...,\nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m=1,...,M$, where ψ_m= $(\psi_{m,1},...,\psi_{m,q_m})$.

For restricted models: For GMAR model: Size $(3M+p-1x1)$ vector θ= $(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,...,\alpha_{M-1})$, where ϕ=$(\phi_1,...,\phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ,ν)= $(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,...,\alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ,ν)= $(\phi_{1,0},...,\phi_{M,0},\phi,\sigma_1^2,...,\sigma_M^2,\alpha_1,.$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1,...,\psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1,...,\phi_p)$ and ψ= $\psi_1,...,\psi_q$.

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

Value

Returns such parameter vector corresponding to the input vector that is the form described in `params` for non-restricted or restricted models (for non-constrained models), and can hence be used just as the parameter vectors of non-constrained models.

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

Description

`reformParameters` 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

```
reformParameters(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$ **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$. Symbol ϕ denotes an AR coefficient, σ^2 a variance, α a mixing weight, and ν a degrees of freedom parameter. 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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"`.

---

reformRestrictedPars *Reform parameter vector with restricted autoregressive parameters to correspond non-restricted parameter vector.*

Description

`reformRestrictedPars` reforms parameter vector with restricted autoregressive parameters to correspond non-restricted parameter vector.

Usage

```
reformRestrictedPars(
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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.

Value

Returns such parameter vector corresponding to the input vector that is the form describted in `params` for non-restricted models (for non-constrained models). Linear constraints are not supported.

`regime_distance` *Calculate "distance" between two regimes*

Description

`regime_distance` scales each regime parameter to the same magnitude and then calculates distance between scaled `regime_pars1` and `regime_pars2`.

Usage

```
regime_distance(regime_pars1, regime_pars2)
```

Arguments

regime_pars1	a numeric vector representing a regime.
regime_pars2	a numeric vector representing another regime, same length as regime_pars1

Value

Returns "distance" between regime_pars1 and regime_pars2. Values are scaled to the same magnitude before calculating the "distance". Read the source code for details.

---

removeAllConstraints *Transform constrained and restricted parameter vector into the regular form*

Description

removeAllConstraints transforms constrained and restricted parameter vector into the regular form.

Usage

```
removeAllConstraints(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where ϕ=$(\phi_1, ..., \phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

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

Value

Returns such parameter vector corresponding to the input vector that is the form described in `params` for non-restricted and non-constrained models.

`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

`simulateGSMAR` *Simulate values from GMAR, StMAR, and G-StMAR processes*

Description

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

Usage

```
simulateGSMAR(gsmar, nsimu, initvalues, ntimes = 1, drop = TRUE)
```

Arguments

`gsmar`	object of class `'gsmar'` created with the function `fitGSMAR` or `GSMAR`.
`nsimu`	a positive integer specifying how many values (ahead from `initvalues`) will be simulated.
`initvalues`	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 not specified, 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 (`nsimu`xM) matrix. Otherwise, returns a list with...

`$sample` a size (`nsimu`x`ntimes`) matrix containing the simulated values.

`$component` a size (`nsimu`x`ntimes`) matrix containing the information from which mixture component each value was generated from.

`$mixing_weights` a size (`nsimu`$xM$$x$`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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

`fitGSMAR`, `GSMAR`, `predict.gsmar`, `add_data`, `condMoments`, `mixingWeights`

Examples

```
# GMAR model
params12 <- c(0.18, 0.93, 0.01, 0.86, 0.68, 0.02, 0.88)
gmar12 <- GSMAR(p=1, M=2, params=params12, model="GMAR")
sim12 <- simulateGSMAR(gmar12, nsimu=500)
ts.plot(sim12$sample)
ts.plot(sim12$component)
ts.plot(sim12$mixing_weights, col=rainbow(2), lty=2)

# FORECASTING EXAMPLE:
# Restricted GMAR model, 10000 sets of simulations with initial values 6 and 6.2.
params22r <- c(1.4, 1.8, 0.8, -0.1, 0.29, 3.18, 0.84)
gmar22r <- GSMAR(p=2, M=2, params=params22r, model="GMAR",
 restricted=TRUE)
sim22r <- simulateGSMAR(gmar22r, nsimu=5, initval=c(6, 6.2), ntimes=10000)
apply(sim22r$sample, 1, median) # Point forecast
apply(sim22r$sample, 1, quantile, probs=c(0.025, 0.975)) # 95% interval
apply(sim22r$mixing_weights, MARGIN=1:2, FUN=median) # mix.weight point forecast
apply(sim22r$mixing_weights, MARGIN=1:2, FUN=quantile,
```

```
 probs=c(0.025, 0.975)) # mix.weight 95% intervals

# G-StMAR model, with initial values
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")
sim12gs <- simulateGSMAR(gstmar12, nsimu=500, initvalues=5:6)
ts.plot(sim12gs$sample)
ts.plot(sim12gs$component)
ts.plot(sim12gs$mixing_weights, col=rainbow(3), lty=2)
```

`sortComponents` *Sort the mixture components of a GMAR, StMAR, or G-StMAR model*

Description

`sortComponents` sorts mixture components of the specified GMAR, StMAR, or G-StMAR model according to the mixing weight parameters when the parameter vector has the "standard/regular form" for restricted or non-restricted models.

Usage

```
sortComponents(
  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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= $(\psi_{m,1}, ..., \psi_{m,q_m})$.

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1})$, where ϕ=$(\phi_1, ..., \phi_M)$.

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= $(\phi_{1,0}, ..., \phi_{M,0}, \phi, \sigma_1^2, ..., \sigma_M^2, \alpha_1, .$

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= $(\psi_1, ..., \psi_q)$.

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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.

Value

Returns a parameter vector sorted according to its mixing weight parameters, described in `params`.

`standardErrors` *Calculate standard errors for estimates of a GMAR, StMAR, or GStMAR model*

Description

`standardErrors` numerically approximates standard errors for the given estimates of GMAR, StMAR, or GStMAR model.

Usage

```
standardErrors(
  data,
  p,
  M,
  params,
```

```
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  custom_h = NULL,
  minval
)
```

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 (2x1) 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: For GMAR model:** Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, α_1, ..., α_{M-1}), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}$, ..., $\phi_{m,p}$), $m = 1, ..., M$. **For StMAR model:** Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_1, ..., ν_M). **For G-StMAR model:** Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, α_1, ..., α_{M-1}, ν_{M1+1}, ..., ν_M). **With linear constraints:** Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}$, ..., ψ_{m,q_m}). **For restricted models: For GMAR model:** Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1}), where ϕ=(ϕ_1, ..., ϕ_M). **For StMAR model:** Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, ..., α_{M-1} **For G-StMAR model:** Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}$, ..., $\phi_{M,0}$,ϕ, σ_1^2, ..., σ_M^2, α_1, . **With linear constraints:** Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= (ψ_1, ..., ψ_q). 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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= $(\phi_1, ..., \phi_p)$ and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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" $\mu_m = \phi_{m,0}/(1-\sum \phi_{i,m})$?

`custom_h` a numeric vector with the same length as `params` specifying the difference 'h' used in finite difference approximation for each parameter separately. 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.

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

Value

Returns approximate standard errors of the parameter values in a numeric vector.

`stmarpars_to_gstmar` *Transform a 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 swicthing the related regimes to be GMAR type.

Usage

```
stmarpars_to_gstmar(
  p,
  M,
  params,
  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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,$\boldsymbol{\phi}_m$, σ_m^2) and $\boldsymbol{\phi}_m$=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors $\boldsymbol{\phi}_m$ with vectors $\boldsymbol{\psi}_m$ and provide a list of constraint matrices **C** that satisfy $\boldsymbol{\phi}_m$=$R_m\boldsymbol{\psi}_m$ for all $m = 1, ..., M$, where $\boldsymbol{\psi}_m$= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where $\boldsymbol{\phi}$=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\boldsymbol{\phi}$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector $\boldsymbol{\phi}$ with vector $\boldsymbol{\psi}$ and provide a constraint matrix C that satisfies $\boldsymbol{\phi}$=$R\boldsymbol{\psi}$, where $\boldsymbol{\psi}$= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m have to be larger than 2.

`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 applied to the autoregressive parameters.

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

For restricted models: a size (pxq) constraint matrix C of full column rank satisfying $\boldsymbol{\phi}$=$C\boldsymbol{\psi}$, where $\boldsymbol{\phi}$= ($\phi_1, ..., \phi_p$) and $\boldsymbol{\psi}$= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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(1, 2, params12, 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`	object of class `'gsmar'` created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# These are long running examples and use parallel computing
fit43t <- fitGSMAR(T10Y1Y, 4, 3, model="StMAR", ncalls=2, seeds=1:2)
fit43t
fit43gst <- stmar_to_gstmar(fit43t)
fit43gst
```

`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`	object of class 'gsmar' created with the function `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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- Virolainen S. 2020. A mixture autoregressive model based on Gaussian and Student's t-distribution. arXiv:2003.05221 [econ.EM].

See Also

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

Examples

```
# Restricted G-StMAR-model with intercept paarametrization
params42gsr <- c(0.11, 0.03, 1.27, -0.39, 0.24, -0.17, 0.03, 1.01, 0.3, 2.03)
gstmar42r <- GSMAR(data=T10Y1Y, p=4, M=c(1, 1), params=params42gsr,
 model="G-StMAR", restricted=TRUE)
summary(gstmar42r)

# Swap to mean parametrization
gstmar42r <- swap_parametrization(gstmar42r)
summary(gstmar42r)
```

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

uGMAR *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. It provides tools for quantile residuals based model diagnostics, forecasting, and simulation, to name a few. Imposing linear constraints to the autoregressive parameters of each regime or restricting them to be the same for all regimes is supported.

Many of the functions documented are not exported but for intended internal use only. The readme file and the vignette are a good place to start.

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

Description

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

Usage

```
uncondMoments(gsmar)
```

Arguments

gsmar	object of class 'gsmar' created with the function 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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- There are currently no published references for the G-StMAR model, but it's a straightforward generalization with theoretical properties similar to the GMAR and StMAR models.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

See Also

Other moment functions: condMoments(), 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")
uncondMoments(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")
uncondMoments(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")
uncondMoments(gstmar12)
```

---

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

Description

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

Usage

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

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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1$, .

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

Symbol ϕ denotes an AR coefficient. Note that regardless of any constraints, the nominal 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

Differs from the function `uncondMoments` in arguments. This function exists for technical reasons only.

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**, 247-266.
- Meitz M., Preve D., Saikkonen P. 2018. A mixture autoregressive model based on Student's t-distribution. arXiv:1805.04010 **[econ.EM]**.
- There are currently no published references for the G-StMAR model, but it's a straightforward generalization with theoretical properties similar to the GMAR and StMAR models.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis. *Springer*.

`warn_dfs` *Warn about large degrees of freedom parameter values*

Description

`warn_dfs` warns if the model contains large degrees of freedom parameter values.

Usage

```
warn_dfs(
  object,
  p,
  M,
  params,
  model = c("GMAR", "StMAR", "G-StMAR"),
  restricted = FALSE,
  constraints = NULL,
  warn_about = c("derivs", "errors")
)
```

Arguments

`object`	an object to be tested
`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 (2x1) 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: For GMAR model: Size $(M(p+3)-1x1)$ vector θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where υ_m= ($\phi_{m,0}$,ϕ_m, σ_m^2) and ϕ_m=($\phi_{m,1}, ..., \phi_{m,p}$), $m = 1, ..., M$.

For StMAR model: Size $(M(p+4)-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_1, ..., \nu_M$).

For G-StMAR model: Size $(M(p+3)+M2-1x1)$ vector (θ, ν)=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}, \nu_{M1+1}, ..., \nu_M$).

With linear constraints: Replace the vectors ϕ_m with vectors ψ_m and provide a list of constraint matrices **C** that satisfy ϕ_m=$R_m\psi_m$ for all $m = 1, ..., M$, where ψ_m= ($\psi_{m,1}, ..., \psi_{m,q_m}$).

For restricted models: For GMAR model: Size $(3M + p - 1x1)$ vector θ= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$), where ϕ=($\phi_1, ..., \phi_M$).

For StMAR model: Size $(4M+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1, ..., \alpha_{M-1}$

For G-StMAR model: Size $(3M+M2+p-1x1)$ vector (θ, ν)= ($\phi_{1,0}, ..., \phi_{M,0}$,$\phi$, $\sigma_1^2, ..., \sigma_M^2, \alpha_1,$.

With linear constraints: Replace the vector ϕ with vector ψ and provide a constraint matrix C that satisfies ϕ=$R\psi$, where ψ= ($\psi_1, ..., \psi_q$).

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 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 parameter α is dropped, and in the case of **StMAR** or **G-StMAR** model, the degrees of freedom parameters ν_m 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 applied to the autoregressive parameters.

For non-restricted models: a list of size (pxq_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 (pxq) constraint matrix C of full column rank satisfying ϕ=$C\psi$, where ϕ= ($\phi_1, ..., \phi_p$) and ψ= $\psi_1, ..., \psi_q$.

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

`warn_about` warn about inaccurate derivatives or standard errors?

Details

Either provide a class 'gsmar' object or specify the model by hand.

Value

Doesn't return anything but throws a warning if any degrees of freedom parameters have value larger than 100.

Index