uGMAR: A Family of Mixture Autoregressive Models in R

Savi Virolainen
University of Helsinki

Abstract

We describe the R package **uGMAR**, which provides tools for estimating and analyzing the Gaussian mixture autoregressive model, the Student's t mixture autoregressive model, and the Gaussian and Student's t mixture autoregressive model. These three models constitute an appealing family of mixture autoregressive models that we call the GSMAR models. The model parameters are estimated with the method of maximum likelihood by running multiple rounds of a two-phase estimation procedure in which a genetic algorithm is used to find starting values for a gradient based method. For evaluating the adequacy of the estimated models, **uGMAR** utilizes so-called quantile residuals and provides functions for graphical diagnostics as well as for calculating formal diagnostic tests. **uGMAR** also enables to simulate from the GSMAR processes and to forecast future values of the process using a simulation-based Monte Carlo method. We illustrate the use of **uGMAR** with the monthly U.S. interest rate spread between the 10-year and 1-year Treasury rates. This vignette was created with the Journal of Statistical Software style, but the manuscript has not been published in the journal.

Keywords: mixture autoregressive model, regime-switching, Gaussian mixture, Student's t mixture.

1. Introduction

Economic time series frequently exhibit non-linear features with the underlying data generating dynamics varying in time, for example, depending on the specific state of the economy. Various types of time series models capable of capturing such regime-switching behavior have been proposed, one of them being the class of mixture models introduced by Le, Martin, and Raftery (1996) and further developed by, among others, Wong and Li (2000, 2001b,a), Glasbey (2001), Lanne and Saikkonen (2003), Kalliovirta, Meitz, and Saikkonen (2015), Meitz, Preve, and Saikkonen (forthcoming), and Virolainen (2020a). Following the recent developments by Kalliovirta *et al.* (2015), Meitz *et al.* (forthcoming), and Virolainen (2020a), we introduce the Gaussian mixture autoregressive (GMAR) model, the Student's t mixture autoregressive (StMAR) model, and the Gaussian and Student's t mixture autoregressive (G-StMAR) model. These three models constitute an appealing family of mixture autoregressive models that we call *the GSMAR models*.

A GSMAR process generates each observation from one of its mixture components, which are either conditionally homoskedastic linear Gaussian autoregressions or conditionally heteroskedastic linear Student's t autoregressions. The mixture component that generates each

observation is randomly selected according to the probabilities determined by the mixing weights that, for a pth order model, depend on the full distribution of the previous p observations. Consequently, the regime-switching probabilities may depend on the level, variability, kurtosis, and temporal dependence of the past observations. The specific formulation of the mixing weights also leads to attractive theoretical properties such as ergodicity and full knowledge of the stationary distribution of $p + 1$ consecutive observations.

This paper describes the R package **uGMAR** providing a comprehensive set of easy-to-use tools for GSMAR modeling, including unconstrained and constrained maximum likelihood (ML) estimation of the model parameters, quantile residual based model diagnostics, simulation from the processes, and forecasting. The emphasis is on estimation, as it can, in our experience, be rather tricky. In particular, due to the endogenously determined mixing weights, the log-likelihood function has a large number of modes, and in large areas of the parameter space, the log-likelihood function is flat in multiple directions. The log-likelihood function's global maximum point is also frequently located very near the boundary of the parameter space. It turns out, however, that such near-the-boundary estimates often maximize the log-likelihood function for a rather technical reason, and it might be more appropriate to consider an alternative estimate based on the largest local maximum point that is clearly in the interior of the parameter space.

The model parameters are estimated by running multiple rounds of a two-phase estimation procedure in which a modified genetic algorithm is used to find starting values for a gradient based variable metric algorithm. Because of the multimodality of the log-likelihood function, some of the estimation rounds may end up in different local maximum points, thereby enabling the researcher to build models not only based on the global maximum point but also on the local ones. The estimated models can be conveniently examined with the `summary` and `plot` methods. For evaluating the models' adequacy, **uGMAR** utilizes quantile residual diagnostics in the framework presented in Kalliovirta (2012), including graphical diagnostics as well as Kalliovirta's (2012) diagnostic tests that take into account uncertainty about the true parameter value. Following Kalliovirta *et al.* (2015) and Meitz *et al.* (forthcoming), forecasting is based on a Monte Carlo simulation method.

Other statistical software implementing the GSMAR models include the **StMAR Toolbox** for Matlab (Meitz, Preve, and Saikkonen 2018). It currently (version 1.0.0) covers the StMAR model of autoregressive orders $p = 1, 2, 3, 4$ and $M = 1, 2, 3$ mixture components, and it contains tools for maximum likelihood estimation, calculation of quantile residuals, simulation, and forecasting. Also the **StMAR Toolbox** estimates the model parameters by using a genetic algorithm to find starting values for a gradient based method, but **uGMAR** takes the procedure of Meitz *et al.* (forthcoming, 2018) further by modifying a genetic algorithm for more efficient estimation. **uGMAR** also has the advantage that it does not impose restrictions on the order the model and it provides a wider variety of tools for analyzing the estimated models; for instance, functions for calculating quantile residual diagnostic tests (Kalliovirta 2012) and plotting the graphs of the profile log-likelihood functions about the estimate. Considering multivariate versions of the GSMAR models, the R package **gmvarkit** (Virolainen 2021) currently (version 1.4.2) accommodates the reduced form and structural Gaussian mixture vector autoregressive models (Kalliovirta, Meitz, and Saikkonen 2016; Virolainen 2020b) and functions similarly to **uGMAR**.

The remainder of this paper is organized as follows. Section 2 introduces the GSMAR models and discusses some of their properties. Section 3 discusses estimation of the model parameters

and illustrates how the GSMAR models can be estimated and examined with **uGMAR**. In Section 4, we describe quantile residuals and demonstrate how they can be utilized to evaluate model adequacy in **uGMAR**. Section 5 shows how the GSMAR models can be built with given parameter values. In Section 6, we first show how to simulate observations from a GSMAR process, and then we illustrate how to forecast future values of a GSMAR process with a simulation-based Monte Carlo method. Section 7 concludes and collects some useful functions in **uGMAR** to a single table for convenience. Finally, Appendix A explains why some maximum likelihood estimates, that are very near the boundary of the parameter space, might be inappropriate and demonstrates that a local maximum point that is clearly in the interior of the parameter space can often be a more reasonable estimate.

Throughout this paper, we use the monthly U.S. interest rate spread between the 10-year and 1-year Treasury rates for the empirical illustrations. We deploy the notation $n_d(\boldsymbol{\mu}, \boldsymbol{\Gamma})$ for the d-dimensional normal distribution with mean $\boldsymbol{\mu}$ and (positive definite) covariance matrix $\boldsymbol{\Gamma}$, and $t_d(\boldsymbol{\mu}, \boldsymbol{\Gamma}, \nu)$ for the d-dimensional t-distribution with mean $\boldsymbol{\mu}$, (positive definite) covariance matrix $\boldsymbol{\Gamma}$, and $\nu > 2$ degrees of freedom. The corresponding density functions are denoted as $n_d(\cdot; \boldsymbol{\mu}, \boldsymbol{\Gamma})$ and $t_d(\cdot; \boldsymbol{\mu}, \boldsymbol{\Gamma}, \nu)$, respectively. By $\mathbf{1}_p = (1, ..., 1)$ $(p \times 1)$, we denote p-dimensional vector of ones.

2. Models

This section introduces the GMAR model (Kalliovirta *et al.* 2015), the StMAR model (Meitz *et al.* forthcoming), and the G-StMAR model (Virolainen 2020a), a family of mixture autoregressive models that we call the GSMAR models. First, we consider the models in a general framework and then proceed to their specific definitions. For brevity, we only give the definition of the more general G-StMAR model but explain how the GMAR and StMAR models are obtained as special cases of it, namely, by taking all the component models to be of either Gaussian or Student's t type.

2.1. Mixture autoregressive models

Let y_t, $t = 1, 2, ...$, be the real valued time series of interest, and let $\mathcal{F}_{t-1}$ denote the σ-algebra generated by the random variables $\{y_{t-j}, j > 0\}$. For a GSMAR model with autoregressive order p and M mixture components, we have

$$y_t = \sum_{m=1}^{M} s_{m,t}(\mu_{m,t} + \sigma_{m,t}\varepsilon_{m,t}), \quad \varepsilon_{m,t} \sim \text{IID}(0, 1), \tag{1}$$

$$\mu_{m,t} = \varphi_{m,0} + \sum_{i=1}^{p} \varphi_{m,i} y_{t-i}, \quad m = 1, ..., M, \tag{2}$$

where $\sigma_{m,t} > 0$ are $\mathcal{F}_{t-1}$-measurable, $\varepsilon_{m,t}$ are independent of $\mathcal{F}_{t-1}$, $\varphi_{m,0} \in \mathbb{R}$, and $s_{1,t}, ..., s_{M,t}$ are unobservable regime variables such that for each t, exactly one of them takes the value one and the others take the value zero. Given the past of y_t, $s_{m,t}$ and $\varepsilon_{m,t}$ are assumed to be conditionally independent, and the conditional probability for an observation to be generated from the mth regime at time t is expressed in terms of ($\mathcal{F}_{t-1}$-measurable) mixing weights $\alpha_{m,t} \equiv \mathsf{P}(s_{m,t} = 1 | \mathcal{F}_{t-1})$ that satisfy $\sum_{m=1}^{M} \alpha_{m,t} = 1$. Furthermore, we assume that for each component model, the autoregressive parameters satisfy the usual stationarity

condition, $1 - \sum_{i=1}^{p} \varphi_{m,i} z^i \neq 0$ for $|z| \leq 1$, which guarantees stationarity of the GSMAR model (Virolainen 2020a, Theorem 1).

The definition (1) and (2) implies that at each t, the observation is generated by a linear autoregression corresponding to some randomly selected (unobserved) mixture component m, and that $\mu_{m,t}$ and $\sigma^2_{m,t}$ can be interpreted as the conditional mean and variance of this component process. In the GMAR model (Kalliovirta *et al.* 2015), the mixture components are conditionally homoskedastic Gaussian autoregressions, whereas in the StMAR model (Meitz *et al.* forthcoming), they are conditionally heteroskedastic Students t autoregressions, while the G-StMAR model (Virolainen 2020a) combines both types of mixture components. The mixing weights, on the other hand, are functions of the preceding p observations.

2.2. The Gaussian and Student's t mixture autoregressive model

In the G-StMAR model, for $m = 1, ..., M_1$ in Equation (1), the terms $\varepsilon_{m,t}$ have standard normal distributions and the conditional variances $\sigma^2_{m,t}$ are constants σ^2_m. For $m = M_1 + 1, ..., M$, the terms $\varepsilon_{m,t}$ follow the t-distribution $t_1(0, 1, \nu_m + p)$ and the conditional variances $\sigma^2_{m,t}$ are defined as

$$\sigma^2_{m,t} = \frac{\nu_m - 2 + (\boldsymbol{y}_{t-1} - \mu_m \mathbf{1}_p)^\top \boldsymbol{\Gamma}_m^{-1} (\boldsymbol{y}_{t-1} - \mu_m \mathbf{1}_p)}{\nu_m - 2 + p} \sigma^2_m, \tag{3}$$

where $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$ $(p \times 1)$, $\nu_m > 2$ is a degrees of freedom parameter, $\sigma^2_m > 0$ is a variance parameter, $\mu_m = \varphi_0 / (1 - \sum_{i=1}^{p} \varphi_{m,i})$ is the stationary mean, and $\boldsymbol{\Gamma}_m$ is the stationary $(p \times p)$ covariance matrix of the mth component process (see Virolainen 2020a, Section 2.1).

This specification leads to a model in which the conditional density function of y_t given its past, $f(\cdot | \mathcal{F}_{t-1})$, is

$$f(y_t | \mathcal{F}_{t-1}) = \sum_{m=1}^{M_1} \alpha_{m,t} n_1(y_t; \mu_{m,t}, \sigma^2_m) + \sum_{m=M_1+1}^{M} \alpha_{m,t} t_1\left(y_t; \mu_{m,t}, \sigma^2_{m,t}, \nu_m + p\right). \tag{4}$$

That is, the first M_1 component processes of the G-StMAR model are homoskedastic Gaussian autoregressions, and the remaining $M_2 \equiv M - M_1$ component processes are heteroskedastic Student's t autoregressions.

In the GMAR model (Kalliovirta *et al.* 2015), all M component processes are Gaussian autoregressions, so its conditional density function is obtained by setting $M_1 = M$ and dropping the second sum in (4). In the StMAR model (Meitz *et al.* forthcoming), all M component processes are Student's t autoregressions, so its conditional density function is obtained by setting $M_1 = 0$ and dropping the first sum in (4). As the component processes of the G-StMAR model coincide with those of the GMAR model and the StMAR model, we often refer to them as *GMAR type* or *StMAR type*, accordingly.

In order to specify the mixing weights, we first define the following function for notational convenience. Let

$$d_m(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m, \nu_m) = \begin{cases} n_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m), & \text{when } m \leq M_1, \\ t_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m, \nu_m), & \text{when } m > M_1, \end{cases} \tag{5}$$

where the p-dimensional densities $n_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m)$ and $t_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m, \nu_m)$ correspond to the stationary distribution of the mth component process (given, for example, in Virolainen 2020a, Equations (2.3) and (2.8)). The mixing weights of the G-StMAR model are defined as

$$\alpha_{m,t} = \frac{\alpha_m d_m(\boldsymbol{y}_{t-1}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m, \nu_m)}{\sum_{n=1}^{M} \alpha_n d_n(\boldsymbol{y}_{t-1}; \mu_n \mathbf{1}_p, \boldsymbol{\Gamma}_n, \nu_n)}, \tag{6}$$

where the parameters $\alpha_1, ..., \alpha_M$ satisfy $\sum_{m=1}^{M} \alpha_m = 1$. The mixing weights of the GMAR model are obtained from (5) and (6) by setting $M_1 = M$, whereas the mixing weights of the StMAR model are obtained by setting $M_1 = 0$.

Because the mixing weights are weighted stationary densities corresponding to the previous p observations, an observation is more likely to be generated from the regime with higher relative weighted likelihood. Moreover, as the mixing weights depend on the full distribution of the previous p observations, the regime-switching probabilities may depend on the level, variability, kurtosis, and temporal dependence of the past observations. This is a convenient property for forecasting, and it also enables the researcher to associate specific characteristics to different regimes.

The specific formulation of the mixing weights also leads to attractive theoretical properties. Specifically, the G-StMAR process $\boldsymbol{y}_t = (y_t, ..., y_{t-p+1})$ $(p \times 1)$, $t = 1, 2, ...$, is ergodic, and it has fully known marginal stationary distribution that is characterized by the density (Virolainen 2020a, Theorem 1; see the proof of Theorem 1 for the stationary distribution of $1, ..., p+1$ consecutive observations)

$$f(\boldsymbol{y}) = \sum_{m=1}^{M_1} \alpha_m n_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m) + \sum_{m=M_1+1}^{M_2} \alpha_m t_p(\boldsymbol{y}; \mu_m \mathbf{1}_p, \boldsymbol{\Gamma}_m, \nu_m). \tag{7}$$

That is, the stationary distribution is a mixture of M_1 p-dimensional Gaussian distributions and M_2 p-dimensional Student's t-distributions with constant mixing weights α_m, $m = 1, ..., M$. For $h = 0, ..., p$, the marginal stationary distribution of $(y_t, ..., y_{t-h})$ is also a mixture of Gaussian and Student's t distributions with constant mixing weights α_m, so the mixing weights parameters α_m can be interpreted as the unconditional probabilities of an observation being generated from the mth component process.

In **uGMAR**, the parameters of the GSMAR models are collected to a $(M(p+3) + M_2 - 1 \times 1)$ vector $\boldsymbol{\theta} \equiv (\boldsymbol{\vartheta}_1, ..., \boldsymbol{\vartheta}_M, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu})$, where $\boldsymbol{\vartheta}_m = (\varphi_{m,0}, \boldsymbol{\varphi}_m, \sigma_m^2)$, $\boldsymbol{\varphi}_m = (\varphi_{m,1}, ..., \varphi_{m,p})$, $m = 1, ..., M$, and $\boldsymbol{\nu} = (\nu_{M_1+1}, ..., \nu_M)$. The parameter α_M is omitted because it is obtained from the restriction $\sum_{m=1}^{M} \alpha_m = 1$, and in the GMAR model, the vector $\boldsymbol{\nu}$ is omitted, as the model does not contain degrees of freedom parameters. The knowledge of the parameter vector is particularly required for building models with given parameter values, which is discussed in Section 5.

3. Estimation

3.1. Log-likelihood function

uGMAR employs the method of maximum likelihood (ML) for estimating the parameters of the GSMAR models. Suppose the observed time series is $y_{-p+1}, ..., y_0, y_1, ..., y_T$ and that the

initial values are stationary. Then, the log-likelihood function of the G-StMAR model takes the form

$$L(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M_1}\alpha_m n_p(\boldsymbol{y}_0;\mu_m\mathbf{1}_p,\boldsymbol{\Gamma}_m) + \sum_{m=M_1+1}^{M}\alpha_m t_p(\boldsymbol{y}_0;\mu_m\mathbf{1}_p,\boldsymbol{\Gamma}_m,\nu_m)\right) + \sum_{t=1}^{T} l_t(\boldsymbol{\theta}), \quad (8)$$

where

$$l_t(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M_1}\alpha_{m,t} n_1(y_t;\mu_{m,t},\sigma_m^2) + \sum_{m=M_1+1}^{M}\alpha_{m,t} t_1\left(y_t;\mu_{m,t},\sigma_{m,t}^2,\nu_m+p\right)\right), \quad (9)$$

and the density functions $n_d(\cdot;\cdot)$ and $t_d(\cdot;\cdot)$ follow the notation described in Section 2.2. Log-likelihood functions of the GMAR model and the StMAR model can be obtained as special cases by setting $M_1 = M$ or $M_1 = 0$, respectively, and dropping the redundant sums.

If stationarity of the initial values seems unreasonable, one can condition on the initial values by dropping the first term on the right hand side of (8) and base the estimation on the resulting conditional log-likelihood function. The ML estimator of a stationary GSMAR model is strongly consistent and has the conventional limiting distribution under the conventional high level conditions as is given in Kalliovirta *et al.* (2015, pp.254-255), Meitz *et al.* (forthcoming, Theorem 2), and Virolainen (2020a, Theorem 2).

3.2. Two-phase estimation procedure

Finding the ML estimate amounts to maximizing the log-likelihood function (8) over a high dimensional parameter space satisfying several constraints. Due to the complexity of the log-likelihood function, finding an analytical solution is infeasible, so numerical optimization methods are required. Following Dorsey and Mayer (1995) and Meitz *et al.* (forthcoming, 2018), **uGMAR** employs a two-phase estimation procedure in which a genetic algorithm is used to find starting values for a gradient based method, which then accurately converges to a nearby local maximum or saddle point. Because of the presence of multiple local maxima, a (sometimes large) number of estimation rounds should be performed to obtain reliable results, for which **uGMAR** makes use of parallel computing to shorten the estimation time.

The genetic algorithm in **uGMAR** is, at core, mostly based on the description by Dorsey and Mayer (1995) but several modifications have been deployed to improve its performance. The modifications include the ones proposed by Patnaik and Srinivas (1994) and Smith, Dike, and Stegmann (1995) as well as further adjustments that take into account model specific issues related to the mixing weights' dependence on the preceding observations. For a more detailed description of the genetic algorithm and its modifications, see Virolainen (2020a, Appendix A). After running the genetic algorithm, the estimation is finalized with a variable metric algorithm (Nash 1990, algorithm 21, implemented by R Core Team 2020) using central difference approximation for the gradient of the log-likelihood function.

3.3. Examples of unconstrained estimation

In this section, we demonstrate how to estimate GSMAR models with **uGMAR** and provide several examples in order to illustrate various frequently occurring situations. In addition to the ordinary estimation, we particularly show how a GSMAR model can be built based on a

local-only maximum point when the ML estimate seems unreasonable (see Appendix A). We also consider the estimation of the appropriate G-StMAR model when the estimated StMAR model contains overly large degrees of freedom estimates (see Virolainen 2020a, Section 4).

For the illustrations, we use the monthly U.S. interest rate spread between the 10-year and 1-year Treasury constant maturity rates, covering the period from 1982 January to 2020 December (468 observations). The series was retrieved from the Federal Reserve Bank of St. Louis database and it is depicted in the top left panel of Figure 1 (Section 3.4). After installing **uGMAR**, the data can be loaded with the following lines of code:

```
R> library("uGMAR")
R> data("M10Y1Y", package = "uGMAR")
```

In **uGMAR**, the GSMAR models are defined as class `gsmar` S3 objects, which can be created with given parameter values using the constructor function `GSMAR` (see Section 5) or by using the estimation function `fitGSMAR`, which estimates the parameters and then builds the model. For estimation, `fitGSMAR` needs to be supplied with a univariate time series and the arguments specifying the model. The necessary arguments for specifying the model include the autoregressive order `p`, the number of mixture components `M`, and `model`, which should be either `"GMAR"`, `"StMAR"`, or `"G-StMAR"`. For GMAR and StMAR models, the argument `M` is a positive integer, whereas for the G-StMAR model it is a length two numeric vector specifying the number of GMAR type regimes in the first element and the number of StMAR type regimes in the second.

Additional arguments may be supplied to `fitGSMAR` in order to specify, for example, whether the exact log-likelihood function should be used instead of the conditional one (`conditional`), whether the model should be parametrized with the intercepts $\varphi_{m,0}$ or the regimewise unconditional means μ_m (`parametrization`), how many estimation rounds should be performed (`ncalls`), and how many central processing unit (CPU) cores should be used in the estimation (`ncores`). Because some of the estimation rounds may end up in local-only maximum points or saddle points, reliability of the estimation results can be improved by increasing the number of estimation rounds. A large number of estimation rounds may be required particularly when the number of mixture components is large, as the surface of the log-likelihood function becomes increasingly more challenging. It is also possible to adjust the settings of the genetic algorithm that is used to find the starting values. The available options are listed in the documentation of the function `GAfit` to which the arguments adjusting the settings will be passed.

The following code fits a StMAR model with autoregressive order $p = 4$ and $M = 2$ mixture components to the monthly interest rate spread using the conditional log-likelihood function and performing 12 estimation rounds with four CPU cores. The argument `seeds` supplies the seeds that initialize the random number generator at the beginning of each call to the genetic algorithm, thereby yielding reproducible results.

```
R> fit42t <- fitGSMAR(M10Y1Y, p = 4, M = 2, model = "StMAR",
+    conditional = TRUE, ncalls = 12, ncores = 4, seeds = 2:13)

Using 4 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=36s
```

```
Results from the genetic algorithm:
The lowest loglik:  143.403
The mean loglik:    160.701
The largest loglik: 172.949
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=06s
Results from the variable metric algorithm:
The lowest loglik:  176.895
The mean loglik:    180.162
The largest loglik: 182.353
Finished!
Warning message:
In warn_dfs(p = p, M = M, params = params, model = model) :
  The model contains overly large degrees of freedom parameter values.
  Consider switching to a G-StMAR model by setting the corresponding
  regimes to be GMAR type with the function 'stmar_to_gstmar'.
```

The progression of the estimation process is reported with a progress bar giving an estimate of the remaining estimation time. Also statistics on the spread of the log-likelihoods are printed after each estimation phase. The progress bars are generated during parallel computing with the package **pbapply** (Solymos and Zawadzki 2020).

The function throws a warning in the above example, because the model contains at least one very large degrees of freedom parameter estimate. Such estimates are warned about, because very large degrees of freedom parameters are redundant in the model and their weak identification might lead to numerical problems, making the approximate standard errors and quantile residual tests often unavailable (Virolainen 2020a, Section 4).

The estimates can be conveniently examined with the `print` method:

```
R> fit42t
```

```
Model:
 StMAR, p = 4, M = 2, #parameters = 15, #observations = 468,
 conditional, intercept parametrization, not restricted, no constraints.

Regime 1
Mix weight: 0.81
Reg mean: 1.87
Var param: 0.04
Df param: 9.76

y = [0.06] + [1.28]y.1 + [-0.36]y.2 + [0.20]y.3 + [-0.15]y.4 + [cond_sd]eps

Regime 2
Mix weight: 0.19
Reg mean: 0.55
Var param: 0.01
```

```
Df param: 10584.18

y = [0.04] + [1.34]y.1 + [-0.59]y.2 + [0.54]y.3 + [-0.36]y.4 + [cond_sd]eps
```

The parameter estimates are reported for each mixture component separately so that the estimates can be easily interpreted. Each regime's autoregressive formula is presented in the form

$$y_t = \varphi_{m,0} + \varphi_{m,1}y_{t-1} + ... + \varphi_{m,p}y_{t-p} + \sigma_{m,t}\varepsilon_{m,t}. \tag{10}$$

The other statistics are listed above the formula, including the mixing weight pameter α_m, the uncondional mean μ_m, the variance parameter σ_m^2, and the degrees freedom parameter ν_m. For GMAR type regimes (if any), the variance parameter σ_m^2 is reported directly in the autoregressive formula, as it is the regime's conditional variance.

The above printout shows that the second regime's degrees of freedom parameter estimate is very large, which might induce numerical problems. However, since a StMAR model with some degrees of freedom parameters tending to infinity coincides with the G-StMAR model with the corresponding regimes switched to GMAR type, one may avoid the problems by switching to the appropriate G-StMAR model (Virolainen 2020a, Section 4). Switching to the appropriate G-StMAR model is recommended also because it removes the redundant degrees of freedom parameters from the model. The function `stmar_to_gstmar` does this switch automatically by first removing the large degrees of freedom parameters and then estimating the G-StMAR model with a variable metric algorithm (Nash 1990, algorithm 21) using the induced parameter vector as the initial value.

To exemplify, the following code switches all the regimes of the StMAR model `fit42t` with a degrees of freedom parameter larger than 100 to GMAR type and then estimates the corresponding G-StMAR model.

```
R> fit42gs <- stmar_to_gstmar(fit42t, maxdf = 100)
```

We use the `summary` method to obtain a more detailed printout of the estimated the G-StMAR model:

```
R> summary(fit42gs, digits = 2)

Model:
 G-StMAR, p = 4, M1 = 1, M2 = 1, #parameters = 14, #observations = 468,
 conditional, intercept parametrization, not restricted, no constraints.

 log-likelihood: 182.35, AIC: -336.71, HQIC: -313.89, BIC: -278.75

Regime 1 (GMAR type)
Moduli of AR poly roots: 1.16, 1.45, 1.45, 1.16
Mix weight: 0.19 (0.09)
Reg mean: 0.55 (NA)
Reg var:  0.14

y = [0.04] + [1.34]y.1 + [-0.59]y.2 + [0.54]y.3 + [-0.36]y.4 + [sqrt(0.01)]eps
```

```
    (0.01)   (0.10)       (0.20)      (0.19)       (0.12)           (0.00)

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.07, 2.02, 2.02, 1.51
Mix weight: 0.81 (NA)
Reg mean: 1.87 (NA)
Var param: 0.04 (0.01)
Df param: 9.76 (4.25)
Reg var:  1.01

y = [0.06] + [1.28]y.1 + [-0.36]y.2 + [0.20]y.3 + [-0.15]y.4 + [cond_sd]eps
    (0.02)   (0.05)       (0.09)      (0.09)       (0.06)

Process mean: 1.62
Process var:  1.11
First p autocors: 0.98 0.96 0.93 0.89
```

In the G-StMAR model, estimates for GMAR type regimes are reported before StMAR type regimes, in a decreasing order according to the mixing weight parameter estimates. As shown above, the model `fit42gs` incorporates one GMAR type regime and one StMAR type regime. Approximate standard errors are given in parentheses under or next to the related estimates. The value `NA` is reported when the related statistic is not parametrized or **uGMAR** was not able to calculate the standard error. In the former case, if one has parametrized the model with intercepts (unconditional means) but wishes to obtain standard errors for the regimewise unconditional means (intercepts), the parametrization may be swapped with the function `swap_parametrization`. The latter case often occurs when the estimates are very close to the boundary of the parameter space or when the model contains extremely large degrees of freedom parameter estimates.

Other reported statistics include the log-likelihood and values of information criteria, the first and second moments of the process, as well as regime specific unconditional means, unconditional variances, and moduli of the roots of the AR polynomials $1-\sum_{i=1}^{p}\varphi_{m,i}z^i$, $m = 1, ..., M$. If some of the moduli are very close to one, the related estimates are near the boundary of the stationarity region. We demonstrate in Appendix A that when such solutions are accompanied with a very small variance parameter estimate, they might not be reasonable estimates and maximize the log-likelihood function for a technical reason only. Consequently, the estimate related to the next-largest local maximum could be considered. This is possible in **uGMAR**, because the estimation function `fitGSMAR` stores the estimates from all estimation rounds so that a GSMAR model can be built based on any one of them, most conveniently with the function `alt_gsmar`. The desired estimation round can be specified either with the argument `which_round` or `which_largest`. The former specifies the round in the estimation order, whereas the latter specifies it in a decreasing order of the log-likelihoods.

To give an example of a case where the estimates are very close the boundary of the stationarity region, we estimate the G-StMAR model directly with the following code.

```
R> fit42gs2 <- fitGSMAR(M10Y1Y, p = 4, M = c(1, 1), model = "G-StMAR",
+    conditional = TRUE, ncalls = 12, ncores = 4, seeds = 1:12)
```

```
Using 4 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=29s
Results from the genetic algorithm:
The lowest loglik:  133.926
The mean loglik:    147.787
The largest loglik: 175.58
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=10s
Results from the variable metric algorithm:
The lowest loglik:  151.91
The mean loglik:    172.875
The largest loglik: 185.558
Finished!
Warning message:
In warn_ar_roots(ret) :
  Regime 1 has near-unit-roots! Consider building a model from the
  next-largest local maximum with the function 'alt_gsmar' by adjusting
  its argument 'which_largest'.
```

The function throws a warning, because the largest found maximum point incorporates a regime that is very close to the boundary of the stationarity region, indicating a potentially inappropriate estimate. We examine the estimates with the `summary` method:

```
R> summary(fit42gs2, digits = 2)

Model:
 G-StMAR, p = 4, M1 = 1, M2 = 1, #parameters = 14, #observations = 468,
 conditional, intercept parametrization, not restricted, no constraints.

 log-likelihood: 185.56, AIC: -343.12, HQIC: -320.30, BIC: -285.16

Regime 1 (GMAR type)
Moduli of AR poly roots: 1.00, 1.00, 1.00, 1.00
Mix weight: 0.02 (0.01)
Reg mean: 0.39 (NA)
Reg var:  0.09

y = [0.63] + [0.21]y.1 + [-0.04]y.2 + [0.21]y.3 + [-1.00]y.4 + [sqrt(0.00)]eps
    (0.01)   (0.01)       (0.02)      (0.01)       (0.00)           (0.00)

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.03, 1.92, 1.92, 1.54
Mix weight: 0.98 (NA)
Reg mean: 0.80 (NA)
Var param: 0.03 (0.01)
Df param: 5.94 (1.97)
```

```
Reg var:  1.65

y = [0.01] + [1.31]y.1 + [-0.40]y.2 + [0.24]y.3 + [-0.17]y.4 + [cond_sd]eps
    (0.01)   (0.05)       (0.08)      (0.08)       (0.05)

Process mean: 0.79
Process var:  1.62
First p autocors: 0.99 0.97 0.95 0.93
```

The summary statistics reveal that there are four near-unit-roots in the GMAR type regime and the variance parameter estimate is very small. Such estimates often occur when there are several regimes in the model and the estimation algorithm is ran a large number of times. If one suspects that the estimate is inappropriate, it is easy to build a model based on the second-largest maximum point that was found in the estimation. Below, the first line of the code builds the model based on the second-largest maximum point, and the second line calls the `summary` method to produce a detailed printout of the model.

```
R> fit42gs3 <- alt_gsmar(fit42gs2, which_largest = 2)
R> summary(fit42gs3, digits = 2)

Model:
 G-StMAR, p = 4, M1 = 1, M2 = 1, #parameters = 14, #observations = 468,
 conditional, intercept parametrization, not restricted, no constraints.

 log-likelihood: 182.35, AIC: -336.71, HQIC: -313.89, BIC: -278.75

Regime 1 (GMAR type)
Moduli of AR poly roots: 1.16, 1.45, 1.45, 1.16
Mix weight: 0.19 (0.09)
Reg mean: 0.55 (NA)
Reg var:  0.14

y = [0.04] + [1.34]y.1 + [-0.59]y.2 + [0.54]y.3 + [-0.36]y.4 + [sqrt(0.01)]eps
    (0.01)   (0.10)       (0.20)      (0.19)       (0.12)           (0.00)

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.07, 2.02, 2.02, 1.51
Mix weight: 0.81 (NA)
Reg mean: 1.87 (NA)
Var param: 0.04 (0.01)
Df param: 9.76 (4.05)
Reg var:  1.01

y = [0.06] + [1.28]y.1 + [-0.36]y.2 + [0.20]y.3 + [-0.15]y.4 + [cond_sd]eps
    (0.02)   (0.05)       (0.09)      (0.09)       (0.06)

Process mean: 1.62
```

```
Process var:  1.11
First p autocors: 0.98 0.96 0.93 0.89
```

The above printout shows that the estimates related to the second-largest local maximum are the same as of the model `fit42gs` (which was estimated based on a StMAR model with a very large degrees of freedom parameter estimate) and that they are clearly inside the stationarity region for all regimes. If also the second-largest maximum point seems unreasonable, a GSMAR model can be built based on the next-largest maximum point by adjusting the argument `which_largest` in the function `alt_gsmar` accordingly.

3.4. Further examination of the estimates

In addition to examining the summary printout, it is often useful to visualize the model by plotting the mixing weights together with the time series and the model's (marginal) stationary density together with a kernel density estimate of the time series. That is exactly what the plot method for GSMAR models does. For instance, the following command creates Figure 1:

```
R> plot(fit42gs)
```

As Figure 1 (the top and bottom left panels) shows, the first regime prevails when the spread takes small values, while the second regime mainly dominates when the spread takes large values. The graph of the model's marginal stationary density (the right panel), on the other hand, shows that the two regimes capture the two modes in the marginal distribution of the spread.

It is also sometimes interesting to examine the time series of (one-step) conditional means and variances of the process along with the time series the model was fitted to. This can be done conveniently with the function `cond_moment_plot`, where the argument `which_moment` should be specified with `"mean"` or `"variance"` accordingly. In addition to the conditional moment of the process, `cond_moment_plot` also displays the conditional means or variances of the regimes multiplied by the mixing weights. Note, however, that the conditional variance of the process is not generally the same as the weighted sum of regimewise conditional variances, as it includes a component that encapsulates heteroskedasticity caused by variation in the conditional mean (see Virolainen 2020a, Equation (2.19)).

The variable metric algorithm employed in the final estimation does not necessarily stop at a local maximum point. The algorithm might also stop at a saddle point or near a local maximum, when the algorithm is not able to increase the log-likelihood, or at any point, when the maximum number of iterations has been reached. In the latter case, the estimation function throws a warning, but saddle points and inaccurate estimates need to be detected by the researcher.

It is well known that in a local maximum point, the gradient of the log-likelihood function is zero, and the eigenvalues of the Hessian matrix are all negative. In a local minimum, the eigenvalues of the Hessian matrix are all positive, whereas in a saddle point, some of them are positive and some negative. Nearly numerically singular Hessian matrices occur when the surface of the log-likelihood function is very flat about the estimate in some directions. This particularly happens when the model contains overly large degrees of freedom parameter

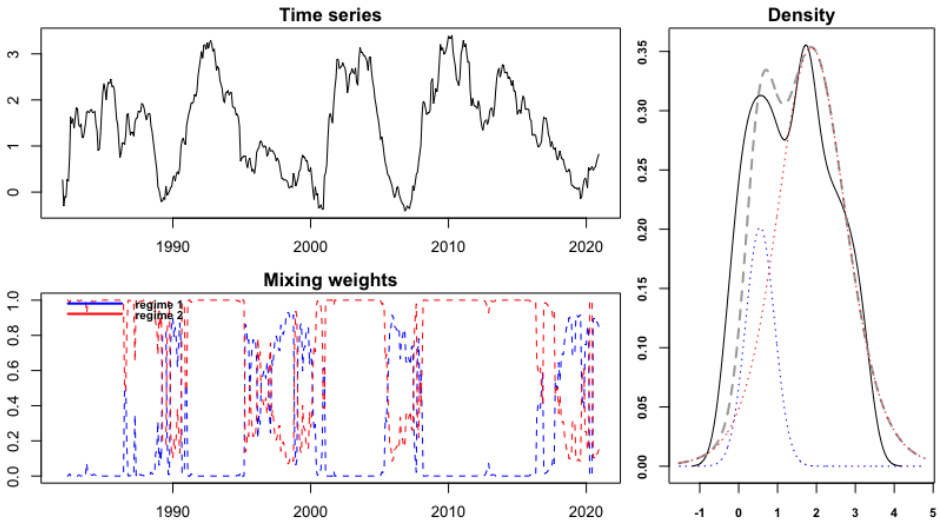

Figure 1: The figure produced by the command `plot(fit42gs)`. On the top left, the monthly spread between the 10-year and 1-year Treasury constant maturity rates, covering the period from 1982 January to 2020 December. On the bottom left, the estimated mixing weights of the G-StMAR model (`fit42gs`) fitted to the interest rate spread (blue dashed line for the first regime and red dashed line for the second regime). On the right, the one-dimensional marginal stationary density of the estimated G-StMAR model (grey dashed line) along with a kernel density estimate of the spread (black solid line) and marginal stationary densities of the regimes multiplied by the mixing weight parameter estimates (blue and red dotted lines).

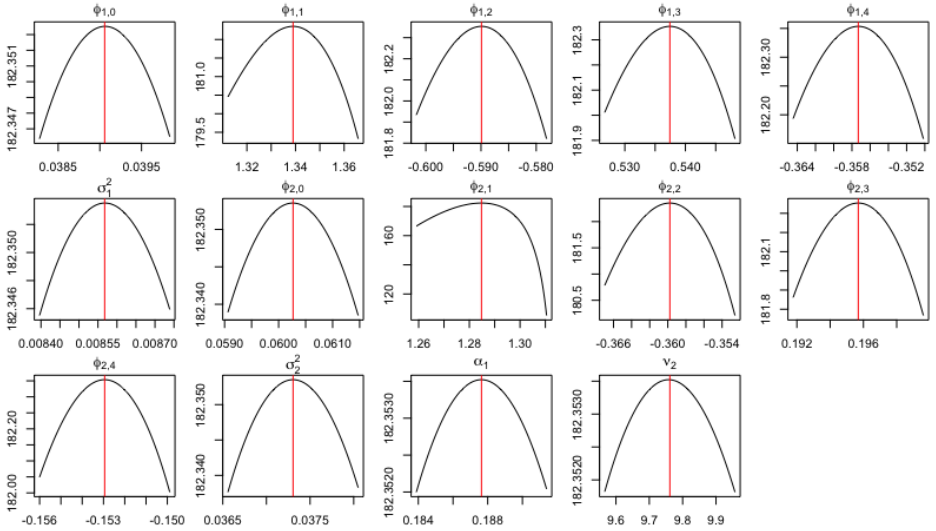

Figure 2: The figure produced by the command `profile_logliks(fit42gs)`. Graphs of the profile log-likelihood functions of the estimated G-StMAR model `fit42gs` with the red vertical lines pointing the estimates.

estimates or the mixing weights $\alpha_{m,t}$ are estimated close to zero for all $t = 1, ..., T$ for some regime m.

uGMAR provides several functions for evaluating whether the estimate is a local maximum point. The function `get_foc` returns the (numerically approximated) gradient of the log-likelihood function evaluated at the estimate, and the function `get_soc` returns eigenvalues of the (numerically approximated) Hessian matrix of the log-likelihood function evaluated at the estimate. The numerical derivatives are calculated using a central difference approximation

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} \approx \frac{f(\boldsymbol{\theta} + \boldsymbol{h}^{(i)}) - f(\boldsymbol{\theta} - \boldsymbol{h}^{(i)})}{2h}, \quad h > 0, \tag{11}$$

where θ_i is the ith element of $\boldsymbol{\theta}$ and $\boldsymbol{h}^{(i)} = (0, ..., 0, h, 0, ..., 0)$ contains h as its ith element. By default, the difference $h = 6 \cdot 10^{-6}$ is used for all parameters except for overly large degrees of freedom parameters, whose partial derivatives are approximated using larger differences. The difference is increased for large degrees of freedom parameters, because the limited precision of the float point presentation induces artificially rugged surfaces to the their profile log-likelihood functions, and the increased differences diminish the related numerical error. On the other hand, as the surface of the profile log-likelihood function is very flat about a large degrees of freedom parameter estimate, large differences work well for the approximation.

For example, the following code calculates the first order condition for the G-StMAR model `fit42gs`:

```
R> get_foc(fit42gs)

 [1]  0.1122667304  0.0443675437  0.0412785752  0.0423330799  0.0452292663
 [6]  0.3538279050 -0.0812537190 -0.0732948896 -0.0755336131 -0.0767990779
[11] -0.0825067588 -0.0117666801 -0.0004755852 -0.000311516
```

and the following code calculates the second order condition:

```
R> get_soc(fit42gs)

 [1] -5.532989e-02 -1.354613e+01 -4.393581e+01 -6.468623e+01 -1.208288e+02
 [6] -1.672930e+02 -2.618770e+02 -8.870701e+02 -2.044356e+03 -4.862843e+03
[11] -4.356473e+04 -5.459504e+04 -2.727102e+05 -5.568025e+05
```

All eigenvalues of the Hessian matrix are negative, which points to a local maximum, but the gradient of the log-likelihood function seems to somewhat deviate from zero. The gradient might be inaccurate, because it is based on a numerical approximation. It is also possible that the estimate is inaccurate, because it is based on approximative numerical estimation, and the estimates are therefore not expected to be exactly accurate. Whether the estimate is a local maximum point with accuracy that is reasonable enough, can be evaluated by plotting the graphs of the profile log-likelihood functions about the estimate. In **uGMAR**, this can be done conveniently with the function `profile_logliks`.

The exemplify, the following command plots the graphs of profile log-likelihood functions of the estimated G-StMAR model `fit42gs`:

```
R> profile_logliks(fit42gs, scale = 0.02, precision = 200)
```

The output is displayed in Figure 2, showing that the estimate's accuracy is reasonable, as changing any individual parameter marginally would not visibly increase the log-likelihood. The argument `scale` can be adjusted to shorten or lengthen the interval shown in the horizontal axis. If one zooms in enough by setting `scale` to a very small value, it can be seen that the estimate is not exactly at the local maximum, but it is so close that moving there would not increase the log-likelihood notably. The argument `precision` can be adjusted to increase the number of points the graph is based on. For faster plotting, it can be decreased, and for more precision, it can be increased.

We have discussed tools that can be utilized to evaluate whether the found estimate is a local maximum with a reasonable accuracy. It is, however, more difficult to establish that the estimate is the global maximum. With **uGMAR**, the best way to increase the reliability that the found estimate is the global maximum, is to run more estimation rounds by adjusting the argument `ncalls` of the estimation function `fitGSMAR`. When a large number of estimation rounds is run (and $M > 1$), `fitGSMAR` often finds peculiar near-the-boundary estimates (see Appendix A) that have extremely spiky profile log-likelihood functions for some parameters and are thus difficult to find. Therefore, it seems plausible that `fitGSMAR` also finds a reasonable ML estimate with a good reliability.

3.5. Examples of constrained estimation

Alternatively to the unconstrained estimation, one may impose linear constraints on the autoregressive (AR) parameters of the model; that is, on $\varphi_{m,1}, ..., \varphi_{m,p}$, $m = 1, ..., M$. **uGMAR** deploys two types of constraints: the AR parameters can be restricted to be the same for all regimes and linear constraints can be applied to each regime separately. In order to impose the former type of constraints, the estimation function simply needs to be supplied with the argument `restricted = TRUE`.

For instance, a GMAR $p = 3$, $M = 2$ model with the AR parameters restricted to be the same in both regimes can be estimated with the following code. The argument `conditional = FALSE` sets the estimation to be based on the exact log-likelihood function and the argument `print_res = FALSE` tells `fitGSMAR` not to the print the spread of the log-likelihoods obtained from each phase of estimation.

```
R> fit32r <- fitGSMAR(M10Y1Y, p = 3, M = 2, model = "GMAR",
+    conditional = FALSE, restricted = TRUE, ncalls = 12,
+    ncores = 4, seeds = 1:12, print_res = FALSE)

Using 4 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=19s
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=01s
Finished!
```

Printout of the model shows the AR parameter estimates are the same in both regimes:

```
R> fit32r

Model:
 GMAR, p = 3, M = 2, #parameters = 8, #observations = 468,
 exact, intercept parametrization, AR parameters restricted, no constraints.

Regime 1
Mix weight: 0.65
Reg mean: 1.61

y = [0.04] + [1.30]y.1 + [-0.33]y.2 + [0.01]y.3 + [sqrt(0.04)]eps

Regime 2
Mix weight: 0.35
Reg mean: 0.77

y = [0.02] + [1.30]y.1 + [-0.33]y.2 + [0.01]y.3 + [sqrt(0.01)]eps
```

The other type constraints in **uGMAR** are of the form

$$\boldsymbol{\varphi}_m = \boldsymbol{C}_m \boldsymbol{\psi}_m, \quad m = 1, ..., M,$$

where $\boldsymbol{C}_m$ is a known $(p \times q_m)$ constraint matrix with full column rank, $\boldsymbol{\psi}_m$ is a $(q_m \times 1)$ parameter vector, and $\boldsymbol{\varphi}_m = (\varphi_{m,1}, ..., \varphi_{m,p})$ contains the AR coefficients of the mth regime. In order to apply the constraints, the estimation function should be supplied with the argument `constraints` containing a list of the constraint matrices $\boldsymbol{C}_m$, $m = 1, ..., M$. For example, to constrain the third AR coefficient of the second regime ($\varphi_{2,3}$) to zero but leaving the first regime unconstrained in a GMAR $p = 3$, $M = 2$ model, we deploy the following list of constraint matrices:

```
R> C_list <- list(diag(3), matrix(c(1, 0, 0, 0, 1, 0), nrow = 3))
R> C_list

[[1]]
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

[[2]]
     [,1] [,2]
[1,]    1    0
[2,]    0    1
[3,]    0    0
```

After setting up the constraints, the constrained model can be estimated as follows:

```
R> fit32c <- fitGSMAR(M10Y1Y, p = 3, M = 2, model = "GMAR",
+    conditional = FALSE, constraints = C_list, ncalls = 12,
+    ncores = 4, seeds = 1:12, print_res = FALSE)

Using 4 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=19s
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=02s
Finished!
```

Printout of the model shows that the third AR parameter estimate of the second regime is zero:

```
R> fit32c

Model:
 GMAR, p = 3, M = 2, #parameters = 10, #observations = 468,
 exact, intercept parametrization, not restricted, linear constraints imposed.

Regime 1
Mix weight: 0.58
Reg mean: 1.17

y = [0.02] + [1.27]y.1 + [-0.21]y.2 + [-0.07]y.3 + [sqrt(0.01)]eps

Regime 2
Mix weight: 0.42
Reg mean: 1.55

y = [0.06] + [1.27]y.1 + [-0.31]y.2 + [0.00]y.3 + [sqrt(0.06)]eps
```

Notice that even when the pth AR coefficient is restricted to zero, the pth lag of that regime is accounted for in the mixing weights (6) and in the case of a StMAR type regime also in the conditional variance (3).

If both types of constraints are applied at the same time, only a single constraint matrix should be supplied (not in a list). Consider a GSMAR model with $p = 2$ and $M = 2$, for example, and suppose the AR coefficients should be restricted to be the same in both regimes and the second AR coefficient ($\varphi_{m,2}$) should be constrained to be the negative of the first coefficient ($\varphi_{m,1}$). Then, the estimation function should be supplied with the arguments `restricted = TRUE` and `constraints = matrix(c(1, -1), nrow=2)`. As demonstrated above, **uGMAR**'s implementation for applying linear constraints is not the most general one, but it makes applying some of the most typical constraints convenient, as the constraint matrices remain small.

4. Quantile residual based model diagnostics

In the GSMAR models, the empirical counterparts of the error terms $\varepsilon_{m,t}$ in Equation (1) cannot be calculated, because the regime that generated each observation is unknown, making the conventional residual based diagnostics unavailable. Therefore, **uGMAR** utilizes so called *quantile residuals*, which are suitable for evaluating adequacy of the GSMAR models. Deploying the framework presented in Kalliovirta (2012), quantile residuals are defined as

$$R_t = \Phi^{-1}(F(y_t|\mathcal{F}_{t-1})), \quad t = 1, 2, ..., T, \tag{12}$$

where $\Phi^{-1}(\cdot)$ is the standard normal quantile function and $F(\cdot|\mathcal{F}_{t-1})$ is the conditional cumulative distribution function of the considered GSMAR process (conditional on the previous observations).

The empirical counterparts of the quantile residuals are calculated by using the parameter estimate and the observed data in (12). For a correctly specified GSMAR model, the empirical counterparts of the quantile residuals based on the ML estimator are asymptotically independent with standard normal distributions (Kalliovirta 2012, Lemma 2.1). Hence, quantile residuals can be used for graphical analysis similarly to the conventional Pearson's residuals.

In **uGMAR**, quantile residuals can be analyzed graphically with the function `diagnostic_plot`, which plots the quantile residual time series, normal quantile-quantile plot, and sample autocorrelation functions of the quantile residuals and squared quantile residuals. If one sets `plot_indstats = TRUE` in the function arguments, `diagnostic_plot` also plots the standardized individual statistics discussed in Kalliovirta (2012, pp. 369-370) with their approximate 95% critical bounds. The individual statistics, which test for remaining autocorrelation or heteroskestacity in specific lags, can be calculated either based on the observed data or based on the simulation procedure proposed by Kalliovirta (2012). In the simulation procedure, the individual statistics' approximate standard errors are based on a (preferably large) sample simulated from the estimated process. According to Kalliovirta's (2012) Monte Carlo study, the simulation procedure may improve size properties of the related tests, but it makes calculation of the statistics computationally more demanding.

The code below creates a diagnostic plot for the G-StMAR model `fit42gs` (estimated in Section 3.3), including the individual statistics based on the observed data and calculated for the first 20 lags:

```
R> diagnostic_plot(fit42gs, nlags = 20, plot_indstats = TRUE)
```

The resulting plot is presented in Figure 3. In order to employ the simulation procedure, one needs to set the length of the simulated sample with the argument `nsimu`. If `nsimu` is not larger than the length of the observed data, the statistics will be based on the observed data. In addition to `diagnostic_plot`, quantile residuals can be graphically examined with the function `quantile_residual_plot`, which plots the quantile residual time series and a histogram.

Analyzing quantile residuals graphically gives an overview of the model's adequacy, but it is often appealing to also carry out a formal testing procedure. Kalliovirta (2012) proposes three specific tests for testing normality, autocorrelation, and conditional heteroskedasticity of the quantile residuals. Kalliovirta's (2012) tests take into account the uncertainty caused by estimation of the parameters and they are shown to perform well in a simulation study (Kalliovirta 2012, Section 4).

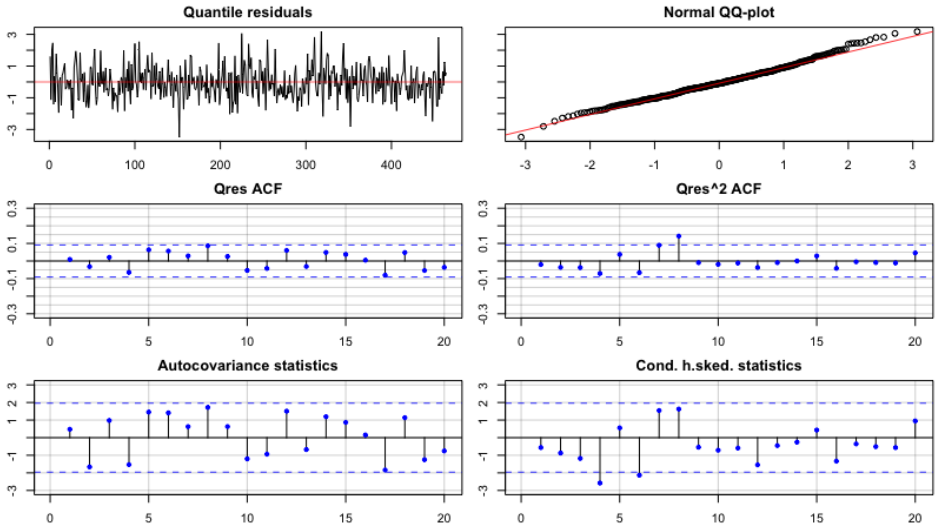

Figure 3: Diagnostic plot for the fitted model `fit42gs` created using the function `diagnostic_plot`. The quantile residual time series (top left), normal quantile-quantile plot (top right), sample autocorrelation functions of the quantile residuals (middle left) and squared quantile residuals (middle right), and the individual autocorrelation (bottom left) and heteroskedasticity (bottom right) statistics discussed in Kalliovirta (2012, pp. 369-370). The blue dashed lines in the sample autocorrelation figures are the $1.96T^{-1/2}$ lines denoting 95% critical bounds for IID-observations, whereas for the Kalliovirta's (2012) individual statistics they are the approximate 95% critical bounds.

In **uGMAR**, the quantile residual tests can be applied with the function `quantile_residual_tests` whose arguments include the model and the numbers of lags to be included in the autocorrelation (`lags_ac`) and heteroskedasticity tests (`lags_ch`). Similarly to the individual statistics discussed in the context of the diagnostic plot, the tests can be based either on the observed data or on the simulation procedure. The simulation procedure can be deployed by setting the argument `nsimu` to be larger than the data length.

The following code calculates the quantile residual tests for the G-StMAR model `fit42gs` by deploying the simulation procedure based on a simulated sample of length 10000 and taking into account $1, 3, 6$, and 12 lags in the autocorrelation and heteroskedasticity tests. By default, the lags for the heteroskedasticity tests are the same as for the autocorrelation tests, so it is enough to set the autocorrelation test lags with the argument `lags_ac`.

```
R> set.seed(1)
R> qrt <- quantile_residual_tests(fit42gs, lags_ac = c(1, 3, 6, 12),
+    nsimu = 10000)

Normality test p-value: 0.089

Autocorrelation tests:
lags | p-value
  1  |  0.475
  3  |  0.018
```

```
  6  |  0.281
  12 |  0.080

Conditional heteroskedasticity tests:
lags | p-value
  1  |  0.584
  3  |  0.140
  6  |  0.003
  12 |  0.000
```

The test results reveal that the model does not seem to adequately capture the conditional heteroskedasticity in the series when taking into account 6 or 12 lags.

uGMAR often fails to calculate the quantile residual tests for GSMAR models with very large degrees of freedom parameter estimates, but the problem can be avoided by switching to the appropriate G-StMAR model with the function `stmar_to_gstmar`, which removes the redundant degrees of freedom parameters (see Virolainen 2020a, Section 4, and Section 3.3 of this paper). Calculation of the tests may also fail when the estimate is very close to the boundary of the parameter space in which case it might be appropriate to consider an estimate from the next-largest local maximum point of the log-likelihood function. To that end, the function `alt_gsmar` can be used as demonstrated in Section 3.3 and in Appendix A.

5. Building a GSMAR model with specific parameter values

The function `GSMAR` facilitates building GSMAR models without estimation, for instance, in order to simulate observations from a GSMAR process with specific parameter values. The parameter vector (of length $M(p+3)+M_2-1$ for unconstrained models) has the form $\boldsymbol{\theta}=(\boldsymbol{\vartheta}_1,...,\boldsymbol{\vartheta}_M,\alpha_1,...,\alpha_{M-1},\boldsymbol{\nu})$ where

$$\boldsymbol{\vartheta}_m=(\varphi_{m,0},\varphi_{m,1},...,\varphi_{m,p},\sigma_m^2),\quad m=1,...,M,\text{ and} \tag{13}$$

$$\boldsymbol{\nu}=(\nu_{M_1+1},...,\nu_M). \tag{14}$$

In the GMAR model (when $M_1=M$), the vector $\boldsymbol{\nu}$ is omitted, as the GMAR model does not contain degrees of freedom parameters. For models with constraints on the autoregressive parameters, the parameter vectors are expressed in a different way. They are only presented in the documentation for brevity, because the hand-specified parameter values can be set to satisfy any constraints as is.

In addition to the parameter vector, `GSMAR` should be supplied with arguments `p` and `M` specifying the order of the model similarly to the estimation function `fitGSMAR` discussed in Sections 3.3 and 3.5. If one wishes to parametrize the model with the regimewise unconditional means (μ_m) instead of the intercepts ($\varphi_{m,0}$), the argument `parametrization` should be set to `"mean"` in which case the intercept parameters $\varphi_{m,0}$ are replaced with μ_m in the parameter vector. By default, **uGMAR** uses intercept parametrization.

To exemplify, we build the GMAR $p=2$, $M=2$ model that is used in the simulation experiment in Appendix A. The model has intercept parametrization and parameter values $\boldsymbol{\vartheta}_1=(0.9,0.4,0.2,0.5)$, $\boldsymbol{\vartheta}_2=(0.7,0.5,-0.2,0.7)$, and $\alpha_1=0.7$. After building the model, we use the `print` method to examine it:

```
R> params22 <- c(0.9, 0.4, 0.2, 0.5, 0.7, 0.5, -0.2, 0.7, 0.7)
R> mod22 <- GSMAR(p = 2, M = 2, params = params22, model = "GMAR")
R> mod22

Model:
 GMAR, p = 2, M = 2, #parameters = 9,
 conditional, intercept parametrization, not restricted, no constraints.

Regime 1
Mix weight: 0.70
Reg mean: 2.25

y = [0.90] + [0.40]y.1 + [0.20]y.2 + [sqrt(0.50)]eps

Regime 2
Mix weight: 0.30
Reg mean: 1.00

y = [0.70] + [0.50]y.1 + [-0.20]y.2 + [sqrt(0.70)]eps
```

It is possible to include data in the models built with `GSMAR` by either providing the data in the argument `data` when creating the model or by adding the data afterwards with the function `add_data`. When the model is supplied with data, the mixing weights, one-step conditional means and variances, and quantile residuals can be calculated and included in the model. The function `add_data` can also be used to update data to an estimated GSMAR model without re-estimating the model.

6. Simulation and forecasting

6.1. Simulation

uGMAR provides the function `simulateGSMAR` for simulating observations from GSMAR processes. `simulateGSMAR` requires the process to be given as a class `gsmar` object, which can be created either by estimating a model with the function `fitGSMAR` or by specifying the parameter values by hand and building the model with the constructor function `GSMAR`. The initial values required to simulate the first p observations can be either set by hand (with the argument `init_values`) or drawn from the stationary distribution of the process (by default). The argument `nsimu` sets the length of the sample path to be simulated.

To give an example, the following code simulates the 500 observations long sample path that is used in the simulation experiment in Appendix A from the GMAR process built in Section 5:

```
R> set.seed(1)
R> mysim <- simulateGSMAR(mod22, nsimu = 500)
```

`simulateGSMAR` returns a list containing the simulated sample path in `$sample`, the mixture component that generated each observation in `$component`, and the mixing weights in `$mixing_weights`.

6.2. Simulation based forecasting

Deriving multiple-steps-ahead point predictions and predictions intervals analytically for the GSMAR models is very complicated, so **uGMAR** employs the following simulation-based method. By using the last p observations of the data up to the date of forecasting as initial values, a large number of sample paths for the future values of the process are simulated. Then, sample quantiles from the simulated sample paths are calculated to obtain prediction intervals, and the median or mean is used for point predictions. A similar procedure is also applied to forecast future values of the mixing weights, which might be of interest because the researcher can often associate specific characteristics to different regimes.

Forecasting is most conveniently done with the `predict` method. The available arguments include the number of steps ahead to be predicted (`n_ahead`), the number sample paths the forecast is based on (`nsimu`), possibly multiple confidence levels for prediction intervals (`pi`), prediction type (`pred_type`), and prediction interval type (`pi_type`). The prediction type can be either `median`, `mean`, or for one-step-ahead forecasts also the exact conditional mean, `cond_mean`. The prediction interval type can be any of `"two-sided"`, `"upper"`, `"lower"`, or `"none"`.

As an example, we use the G-StMAR $p = 4, M_1 = 1, M_2 = 1$ model fitted to the monthly interest rate spread in Section 3.3 to forecast the spread 12 months ahead, i.e., for the year 2021. The point prediction is based on median and 10000 simulated future sample paths, and the two-sided prediction intervals are calculated for the confidence levels 0.95 and 0.80.

```
R> set.seed(1)
R> mypred <- predict(fit42gs, n_ahead = 12, nsimu = 10000, pi = c(0.95, 0.8),
+    pred_type = "median", pi_type = "two-sided")
R> mypred
```

```
Prediction by median, two-sided prediction intervals with levels 0.95, 0.8.
Forecast 12 steps ahead, based on 10000 simulations.

   0.025  0.1 median  0.9 0.975
1   0.66 0.74   0.87 1.01  1.10
2   0.54 0.66   0.89 1.13  1.32
3   0.46 0.62   0.90 1.23  1.48
4   0.36 0.55   0.91 1.33  1.65
5   0.25 0.49   0.91 1.44  1.83
6   0.17 0.43   0.90 1.54  2.00
7   0.08 0.38   0.91 1.64  2.15
8   0.02 0.33   0.91 1.73  2.25
9  -0.02 0.29   0.91 1.81  2.35
10 -0.07 0.26   0.91 1.88  2.45
11 -0.09 0.24   0.92 1.95  2.58
12 -0.13 0.22   0.93 2.01  2.65

 Point forecasts and prediction intervals for mixing weights can be obtained
 with $mix_pred and $mix_pred_ints, respectively.
```

The `predict` method plots the results by default but this can be also avoided by setting

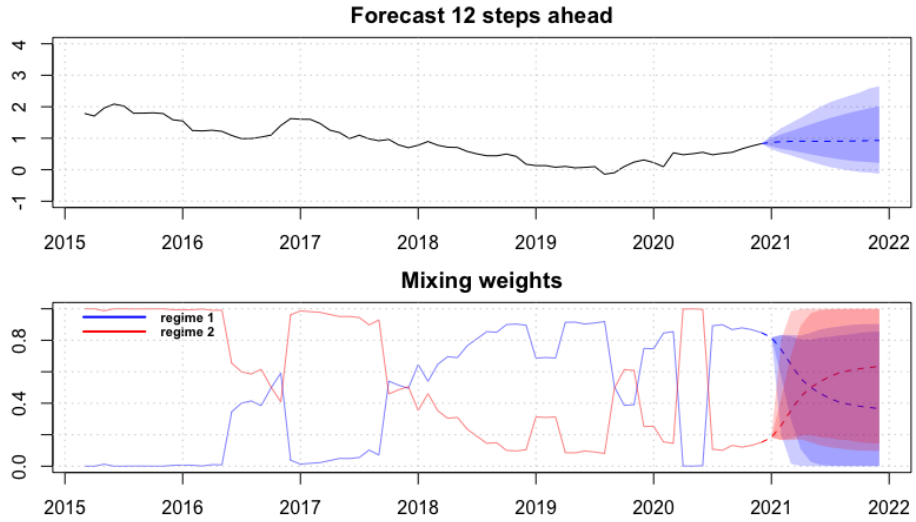

Figure 4: Figure created by the `predict` method for the G-StMAR model `fit42gs`. Twelve-months-ahead point prediction for the monthly interest rate spread (top) and the model's mixing weights (bottom) together with several preceding observations and prediction intervals with confidence levels 0.95 (outer interval) and 0.80 (inner interval).

`plot_res = FALSE` in the arguments. The figure created by the above example is presented in Figure 4.

7. Summary

Mixture autoregressive models are useful for analyzing time series that exhibit non-linear, regime-switching features. The GMAR model, the StMAR model, and the G-StMAR model constitute an appealing family of such models, the GSMAR models, with attractive theoretical and practical properties. This paper introduced the R package **uGMAR** providing a comprehensive set of easy-to-use tools for GSMAR modeling, including unconstrained and constrained maximum likelihood estimation of the model parameters, quantile residual based model diagnostics, simulation, forecasting, and more. For convenience, we have collected some useful functions in **uGMAR** to Table 1, which contains also functions not mentioned in the text.

The model parameters are estimated with the method of maximum likelihood by employing a two-phase procedure, which uses a genetic algorithm to find starting values for a variable metric algorithm. Notably, due to the endogenously determined mixing weights, the maximum likelihood estimate is occasionally found very close to the boundary of the stationarity region of some regimes. We explained in an Appendix why such estimates might be inappropriate and showed how a GSMAR model can be built based on an alternative estimate related to the next-largest local maximum point.

Related to	Name	Description
Estimation	`fitGSMAR`	Estimate a GSMAR model.
	`alt_gsmar`	Build a GSMAR model based on results from any estimation round.
	`stmar_to_gstmar`	Estimate a G-StMAR model based on a StMAR (or G-StMAR) model with large degrees of freedom parameters.
	`iterate_more`	Run more iterations of the variable metric algorithm for a preliminary estimated GSMAR model.
Estimates	`summary` (method)	Detailed printout of the estimates.
	`plot` (method)	Plot the series with the estimated mixing weights and kernel density estimate of the series with the stationary density of the model.
	`get_foc`	Calculate numerically approximated gradient of the log-likelihood function evaluated at the estimate.
	`get_soc`	Calculate eigenvalues of numerically approximated Hessian of the log-likelihood function evaluated at the estimate.
	`profile_logliks`	Plot the graphs of the profile log-likelihood functions.
	`cond_moment_plot`	Plot the model implied one-step conditional means or variances.
Diagnostics	`quantile_residual_tests`	Calculate quantile residual tests.
	`diagnostic_plot`	Plot quantile residual diagnostics.
	`quantile_residual_plot`	Plot quantile residual time series and histogram.
Forecasting	`predict` (method)	Forecast future observations and mixing weights of the process.
Simulation	`simulateGSMAR`	Simulate from a GSMAR process.
Create model	`GSMAR`	Construct a GSMAR model based on specific parameter values.
Hypothesis testing	`LR_test`	Calculate likelihood ratio test.
	`Wald_test`	Calculate Wald test.
Other	`add_data`	Add data to a GSMAR model
	`swap_parametrization`	Swap to mean/intercept parametrization

Table 1: Some useful functions in **uGMAR** sorted according to their usage. The note "method" in brackets after the name of a function signifies that it is an S3 method for a class `gsmar` object (often generated by the function `fitGSMAR` or `GSMAR`).

Computational details

The results in this paper were obtained using R 4.0.2 and **uGMAR** 3.3.1 package running on MacBook Pro 13", 2020, with Intel Core i5-8257U 1.40Ghz processor and 8 Gt 2133 Mhz LPDDR3 RAM.

uGMAR takes use of the R package **Brobdingnag** (Hankin 2007) to handle values extremely close to zero in the evaluation of the first term of the exact log-likelihood function (8). The package **gsl** (Hankin, Clausen, and Murdoch 2006) is utilized to calculate some of the quantile residuals (12) with a hypergeometric function. In order to improve computational efficiency in the numerical estimation procedure, the formula proposed by Galbraith and Galbraith (1974) is utilized to directly compute the inverses of the covariance matrices $\mathbf{\Gamma}_m$, $m = 1, ..., M$, (which appear in Equations (3), (5), (6), and in the first term of (8)), as only the inverses are required for calculating the quantities in the log-likelihood function. Finally, the algorithm proposed by Monahan (1984) is employed to generate random stationary autoregressive coefficients in the genetic algorithm.

Acknowledgments

The author thanks Markku Lanne, Mika Meitz, and Pentti Saikkonen from comments and discussions, which helped to improve this paper substantially. The author also thanks Academy of Finland for financing the project (grant 308628).

References

Dorsey R, Mayer W (1995). "Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features." *Journal of Business and Economic Statistics*, **13**(1), 53–66. doi:10.1080/07350015.1995.10524579.

Galbraith J, Galbraith R (1974). "On the Inverses of Some Patterned Matrices Arising in the Theory of Stationary Time Series." *Journal of Applies Probability*, **11**(1), 63–71. doi:10.2307/3212583.

Glasbey C (2001). "Non-linear Autoregressive Time Series with Multivariate Gaussian Mixtures as Marginal Distributions." *Journal of Royal Statistical Society: Series C*, **50**(2), 143–154. doi:10.1111/1467-9876.00225.

Hankin R (2007). "Very large numbers in R: Introducing package Brobdingnag." *R News*, **7**(3).

Hankin R, Clausen A, Murdoch D (2006). "Special Functions in R: Introducing the gsl Package." *R News*, **6**(4).

Kalliovirta L (2012). "Misspecification Tests Based on Quantile Residuals." *The Econometrics Journal*, **15**(2), 358–393. doi:10.1111/j.1368-423X.2011.00364.x.

Kalliovirta L, Meitz M, Saikkonen P (2015). "A Gaussian Mixture Autoregressive Model for Univariate Time Series." *Journal of Time Series Analysis*, **36**(2), 247–266. doi:10.1111/jtsa.12108.

Kalliovirta L, Meitz M, Saikkonen P (2016). "Gaussian Mixture Vector Autoregression." *Journal of Econometrics*, **192**(2), 465–498. doi:10.1016/j.jeconom.2016.02.012.

Lanne M, Saikkonen P (2003). "Modeling the U.S. Short-Term Interest Rate by Mixture Autoregressive Processes." *Journal of Financial Econometrics*, **1**(1), 96–125. doi:10.1093/jjfinec/nbg004.

Le N, Martin R, Raftery A (1996). "Modeling Flat Stretches, Bursts, and Outliers in Time Series Using Mixture Transition Distribution Models." *Journal of the American Statistical Association*, **91**(436), 1504–1515. doi:10.2307/2291576.

Meitz M, Preve D, Saikkonen P (2018). *StMAR Toolbox: A MATLAB Toolbox for Student's t Mixture Autoregressive Models*. doi:10.2139/ssrn.3237368.

Meitz M, Preve D, Saikkonen P (forthcoming). "A Mixture Autoregressive Model Based on Student's t-Distribution." *Communications in Statistics - Theory and Methods*. doi:10.1080/03610926.2021.1916531.

Monahan J (1984). "A Note on Enforcing Stationarity in Autoregressive-Moving Average Models." *Biometrika*, **71**(2), 403–404. doi:10.1093/biomet/71.2.403.

Nash J (1990). *Compact Numerical Methods for Computers. Linear Algebra and Function Minimization*. 2nd edition. Adam Hilger, Bristol and New York. doi:10.1201/9781315139784.

Patnaik L, Srinivas M (1994). "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms." *Transactions on Systems, Man and Cybernetics*, **24**(4), 656–667. doi:10.1109/21.286385.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Smith R, Dike B, Stegmann S (1995). "Fitness Inheritance in Genetic Algorithms." *Proceedings of the 1995 ACM symbosium on Applied Computing*, pp. 345–350. doi:10.1145/315891.316014.

Solymos P, Zawadzki Z (2020). *pbapply: Adding Progress Bar to '*apply' Functions*. R package version 1.4-3, URL https://CRAN.R-project.org/package=pbapply.

Virolainen S (2020a). "A Mixture Autoregressive Model Based on Gaussian and Student's t-distributions." *Unpublished working paper, available as arXiv:2003.05221*. URL https://arxiv.org/abs/2003.05221.

Virolainen S (2020b). "Structural Gaussian Mixture Vector Autoregressive Model." *Unpublished working paper, available as arXiv:2007.04713*. URL https://arxiv.org/abs/2007.04713.

Virolainen S (2021). *gmvarkit: Estimate Gaussian Mixture Vector Autoregressive Model*. R package version 1.4.2, URL https://CRAN.R-project.org/package=gmvarkit.

Wong C, Li W (2000). "On a Mixture Autoregressive Model." *Journal of the Royal Statistical Society*, **62**(1), 95–115. doi:10.1093/biomet/asp031.

Wong C, Li W (2001a). "On a Logistic Mixture Autoregressive Model." *Biometrika*, **88**(3), 833–846. doi:10.1093/biomet/88.3.833.

Wong C, Li W (2001b). "On a Mixture Autoregressive Conditional Heteroskedastic Model." *Journal of the American Statistical Association*, **96**(455), 982–995. doi:10.2307/2670244.

A. Simulation experiment

This simulation experiment demonstrates why the log-likelihood function's global maximum point, that is found very near the boundary of the parameter space, might not be a reasonable estimate and why it might be more appropriate to consider a local-only maximum point that is clearly in the interior of the parameter space. We generated 500 observations from a GMAR $p = 2$, $M = 2$ process with the parameter values given in the first row of Table 2 ($\boldsymbol{\theta}$) and initial values generated from the stationary distribution of the process. This model is built with **uGMAR** as an example in Section 5, and the sample path is generated as an example in Section 6.1.

We estimated a GMAR $p = 2$, $M = 2$ model to the generated sample based on the exact log-likelihood function by performing 100 estimation rounds using the following code (output is omitted for brevity):

```
R> fit22 <- fitGSMAR(mysim$sample, p = 2, M = 2, model = "GMAR",
+    conditional = FALSE, ncalls = 100, ncores = 4, seeds = 1:100)
```

The obtained estimates are reported on the second row of Table 2 ($\hat{\boldsymbol{\theta}}_1$) together with the moduli of each regime's AR polynomial's $(1-\sum_{i=1}^{p}\varphi_{m,i}z^i)$ roots. The modulus of the ith root in the mth regime is denoted by the symbol $\xi_{m,i}$. The stationarity condition requires that all the moduli are strictly greater than one, so the second regime is very close to the boundary of the stationarity region (both roots are approximately 1.000006). Also the variance parameter σ_2^2 is close to its lower bound zero (it is approximately $6 \cdot 10^{-6}$).

These estimates produce a large log-likelihood, because the second regime's very small conditional variance makes the related density function in the term $l_t(\boldsymbol{\theta})$ (9) to take large values near its mean, and the strong conditional mean targets individual observations there. This is illustrated in Figure 5 (bottom panel), where the terms $l_t(\boldsymbol{\theta})$ are presented (green solid line) together with the second regime's related weighted densities $\alpha_{2,t}n_1(y_t;\mu_{2,t},\sigma_2^2)$ (red dotted line). The black "X"-symbols denote the points where the second regime's conditional mean deviates from the corresponding observation by less than 0.005. Evidently, the second regime contributes to the log-likelihood function only in the individual points where both, the terms $l_t(\boldsymbol{\theta})$ and the scaled densities $\alpha_{2,t}n_1(y_t;\mu_{2,t},\sigma_2^2)$, take large values due to the observation being close to the mean of the second regime's spikelike conditional density function. Because the scaled densities take large enough values in those individual points, the log-likelihood is larger for this kind of estimate than for a reasonable estimate.

The top panel of Figure 5 presents the true mixing weights of the GMAR process's second regime (black solid line) together with the mixing weights based on the estimate $\hat{\boldsymbol{\theta}}_1$ (red dashed line). As the figure shows, the estimated mixing weights are spiky and have no resemblance to the true mixing weights. Although the true mixing weights can be spiky for some GSMAR processes, spiking mixing weights are also typical for potentially inappropriate near-the-boundary estimates.

This kind of near-the-boundary estimates are often found when a subset of the regimes explains the variation in the series reasonably well, leaving some of the regimes available for targeting individual observations with very small conditional variance and very strong conditional mean. As such estimates seem to maximize the log-likelihood function for a technical reason, and not necessarily because they represent a good guess for the true parameter value, it might be appropriate to consider an alternative estimate related to the next-largest local

	$\varphi_{1,0}$	$\varphi_{1,1}$	$\varphi_{1,2}$	σ_1^2	$\varphi_{2,0}$	$\varphi_{2,1}$	$\varphi_{2,2}$	σ_2^2	α_1	$\xi_{1,1}$	$\xi_{1,2}$	$\xi_{2,1}$	$\xi_{2,2}$
$\boldsymbol{\theta}$	0.90	0.40	0.20	0.50	0.70	0.50	−0.20	0.70	0.70	1.45	3.45	2.24	2.24
$\hat{\boldsymbol{\theta}}_1$	0.59	0.55	0.11	0.62	7.85	−1.67	−1.00	0.00	0.99	1.43	6.48	1.00	1.00
$\hat{\boldsymbol{\theta}}_2$	1.16	0.39	0.08	0.54	0.77	0.35	−0.17	0.53	0.63	1.86	6.90	2.42	2.42

Table 2: On the first row, the true parameter values of the GMAR $p = 2$, $M = 2$ process that generated the sample path used in the simulation experiment. On the second row, the estimates that maximized the log-likelihood function based 100 estimation rounds. On the third row, the estimates from the largest such log-likelihood function's maximum point that is not very near the boundary of the stationarity region. In each row after the estimates or parameter values, the moduli of the related AR polynomial's roots are presented.

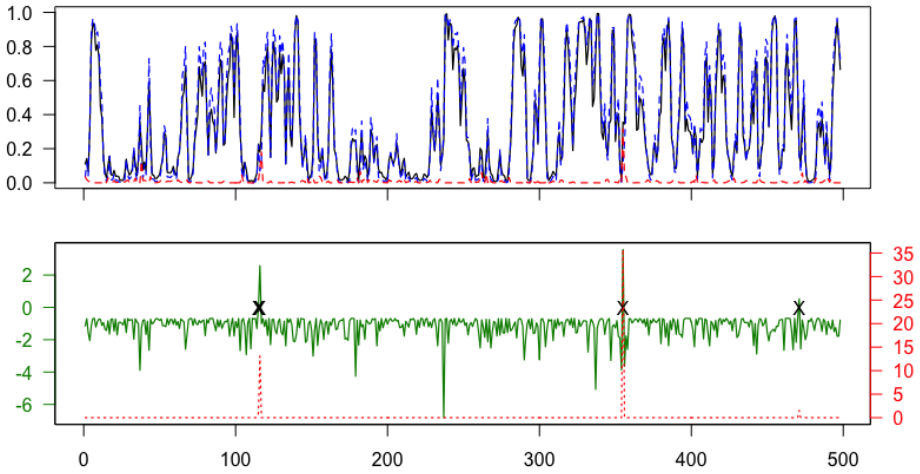

Figure 5: On the top, the GMAR $p = 2$, $M = 2$ process's second regime's true mixing weights (black solid line), the mixing weights based on the estimate $\hat{\boldsymbol{\theta}}_1$ in the second row of Table 2 (red dashed line), and the mixing weights based on the estimate $\hat{\boldsymbol{\theta}}_2$ in the third row of Table 2 (blue dashed line). On the bottom, the terms (9) from the second term of the log-likelihood function (8) (green solid line) and the second regime's densities in the terms (9) multiplied by the estimated mixing weights (blue dotted line), i.e., $\alpha_{2,t}n_1(y_t; \mu_{2,t}, \sigma_2^2)$, both based on the estimate $\hat{\boldsymbol{\theta}}_1$. The "X"-symbols denote the points where the second regime's conditional mean for the model based on estimate $\hat{\boldsymbol{\theta}}_1$ deviates from the corresponding observation by less than 0.005.

maximum point. To exemplify, we build a model based on the largest local maximum point that is clearly in the interior of the parameter space. In our estimation based on 100 rounds of the two-phase procedure, such an estimate is found at the point that induced the third largest log-likelihood, and it is obtained as follows:

```
R> fit22_alt <- alt_gsmar(fit22, which_largest = 3)
```

The corresponding estimate is presented on the third row of Table 2 ($\hat{\boldsymbol{\theta}}_2$). This local maximum point is substantially closer to the true parameter value in the second regime. The resemblance to the true parameter value is also highlighted in Figure 5 (top panel), where the second regime's estimated mixing weights (blue dashed line) are presented together with the true mixing weights (black solid line).

Finally, note that the estimate $\hat{\boldsymbol{\theta}}_1$ presented in Table 2 is not the accurate maximum likelihood estimate, which can be noticed by examining graphs of the related profile log-likelihood functions with the command `profile_logliks(fit22)` (not shown). The numerical estimation using numerical approximation for the gradient of the log-likelihood function can be inaccurate near the boundary of a multidimensional parameter space subject to several constraints. Consequently, other similar near-the-boundary points that induce larger log-likelihood than $\hat{\boldsymbol{\theta}}_1$ can be found by running more estimation rounds. It should also be noted that sometimes the estimate is near the boundary of the stationarity region because the series is very persistent, and being near the boundary does not hence necessarily imply that the MLE is inappropriate.

Affiliation:

Savi Virolainen
Faculty of Social Sciences
University of Helsinki
P. O. Box 17, FI-0014 University of Helsinki, Finland
E-mail: savi.virolainen@helsinki.fi