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.

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

1. Introduction

A popular method for modeling univariate time series is to employ a linear autoregressive (AR) model that assumes the process to be generated by a weighted sum of the preceding p observations, an intercept term, and a random error. The error process is often assumed to be serially uncorrelated with zero mean and constant variance. This encompasses conditionally homoskedastic processes, such as independent and identically distributed (IID) processes, as well as conditionally heteroskedastic processes, such as autoregressive conditional heteroskedasticity (ARCH) processes (Engle 1982) and generalized autoregressive conditional heteroskedasticity (GARCH) processes (Bollerslev 1986).

Several R packages accommodate linear AR modeling with various types of error processes. The R package **forecast** (Hyndman, Athanasopoulos, Bergmeir, Caceres, Chhay, O'Hara-Wild, Petropoulos, Razbash, Wang, and Yasmeen 2021), for instance, accommodates estimation of AR models with seasonal components. The R package **fGarch** (Wuertz, Setz, Chalabi, Boudt, Chausse, and Miklovac 2020), on the other hand, allows to estimate AR models with ARCH and GARCH errors following various distributions, including normal distribution, Student's t-distribution, generalized error distribution, and their skewed versions. A more comprehensive set of error processes are provided in the popular R package **rugarch** (Ghalanos 2020). It accommodates a rich set of different GARCH processes with several error distributions,

including the regular and skewed versions of normal, t-, and generalized error distributions, as well as generalized hyperbolic normal and inverse Gaussian distributions, to name a few.

A linear AR model with potentially skewed GARCH errors can often filter the autocorrelation and conditional heteroskedasticity from the series very well. But in some cases, it cannot adequately capture all the relevant characteristics of the series, including shifts in the mean or volatility, and changes in the dynamics of the process. Such nonlinear features frequently occur in economic time series when the underlying data generating dynamics vary in time, for example, depending 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 (2021), and Virolainen (2021b). Following the recent developments by Kalliovirta *et al.* (2015), Meitz *et al.* (2021), and Virolainen (2021b), we consider 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 their 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.* (2021), 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.* (2018, 2021) further by modifying a genetic algorithm for more efficient estimation. **uGMAR** also has the advantage that it does not impose restrictions on the order of 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.

The R package **gmvarkit** (Virolainen 2018) functions similarly to **uGMAR** and accommodates multivariate versions of the GSMAR models, including structural models with statistically identified shocks. These models include the (structural) Gaussian mixture vector autoregressive model (Kalliovirta, Meitz, and Saikkonen 2016; Virolainen 2021c), the (structural) Student's t mixture vector autoregressive model (Virolainen 2021a), and the (structural) Gaussian and Student's t mixture vector autoregressive model (Virolainen 2021a). The R package **mixAR** (Boshnakov and Ravagli 2021), in turn, allows frequentist and Bayesian estimation of mixture (vector) autoregressive models with constant mixing weights (e.g., Wong and Li 2000; Fong, Li, Yau, and Wong 2007) and various error distributions.

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 model selection. It also illustrates how the GSMAR models can be estimated and examined with **uGMAR**, and how parameter constraints can be tested. 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. 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. Finally, Appendix B derives closed form expressions for the quantile residuals of the GSMAR models.

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.* 2021), and the G-StMAR model (Virolainen 2021b), 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 models (Virolainen 2021b, 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.* 2021), they are conditionally heteroskedastic Students t autoregressions, while the G-StMAR model (Virolainen 2021b) combines both types of mixture components. The mixing weights 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 (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

$$\sigma^2_{m,t} = \frac{\nu_m - 2 + (\boldsymbol{y}_{t-1} - \mu_m \mathbf{1}_p)' \mathbf{\Gamma}_m^{-1} (\boldsymbol{y}_{t-1} - \mu_m \mathbf{1}_p)}{\nu_m - 2 + p} \sigma^2_m, \tag{3}$$

where $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$ $(p \times 1)$, $\nu_m > 2$ is a degrees of freedom parameter, $\sigma_m^2 > 0$ 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 2021b, 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

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

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.* 2021), 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 2021b, 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 2021b, 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 and model selection

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

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

where

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

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.* (2021, Theorem 3), and Virolainen (2021b, 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.* (2021, 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 (2021b, 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 2021) using central difference approximation for the gradient of the log-likelihood function.

3.3. Model selection

Before illustrating with examples how the GSMAR models can be estimated with **uGMAR**, it is helpful to first briefly discuss the problem of model selection. Finding a suitable GSMAR model involves several selections: one needs to choose the type of the model (GMAR, StMAR, or G-StMAR), the autoregressive order p, and the number of mixture components M (in the G-StMAR model, the number of GMAR type regimes M_1 and the number of StMAR type regimes M_2). Following Kalliovirta *et al.* (2015, Section 3.1), we suggest starting the model selection by first considering linear AR models, and then building up to the more complex regime-switching models if the linear models are found inadequate. After finding a suitable GSMAR model, simplifications obtained by parameter restrictions can be considered (constrained estimation is discussed in Section 3.6, testing the constraints in Section 3.7, and diagnostics checks for evaluating the adequacy of the model in Section 4).

When selecting the type of the GSMAR model, it is useful to take into account the features of the different types of models. The GMAR model incorporates linear Gaussian AR processes as its mixture components and can flexibly model changes in the conditional mean. But as its component processes are conditionally homoskedastic, it can capture changes in the conditional variance only through the regime-switching dynamics. The StMAR model, on the other hand, incorporates ARCH type conditional heteroskedasticity within each regime with the conditional variance (3), and can thereby account for stronger forms of conditional heteroskedasticity. In the StMAR model, the autoregressive order p is also the lag order of the ARCH type conditional variance. The conditional variance (3), however, depends on the past observations through the same parameters as the conditional mean (2), which can restrictive when the regime-specific conditional mean is strong but conditional variance is weak[1] (or vice

[1]By strong (weak) conditional variance or mean, we mean strong (weak) dependence on the preceding observations.

versa). It may therefore be worthwhile to first try whether the simpler GMAR model can adequately explain the characteristics of the series.

If the conditional variance is constant in some regimes but time-varying in other regimes, the G-StMAR model can be employed, as it contains both conditionally homoskedastic GMAR type regimes and conditionally heteroskedastic StMAR type regimes. For choosing the number of GMAR and StMAR type regimes in the G-StMAR model, we suggest following the strategy of Virolainen (2021b, Section 4) and first finding a suitable StMAR model. If the estimated StMAR model contains overly large degrees of freedom parameter estimates, those regimes should be switched to GMAR type by estimating the appropriate G-StMAR model (this is discussed in more detail Section 3.4).

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. After installing **uGMAR**, the data can be loaded with the following lines of code:

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

For finding the suitable type and order of the model, it is often useful to plot several figures illustrating the statistical properties of the series. A time series plot can be examined to obtain an overall perception of series, and to investigate whether there seem to be apparent changes in the dynamics of series, or shifts in the mean or volatility that would indicate a possible presence of multiple regimes.

The time series plot of the interest rate spread `M10Y1Y` is shown in the top left panel of Figure 2 (in Section 3.5). It shows that the process consistently produces consecutive observations of the same magnitude, which are then followed by a transition to another magnitude. There thus appears to be shifts in the mean of the process and the changes are occasionally rapid.

A non-parametric estimate of the density function, such as a kernel density estimate, can be examined to evaluate whether the marginal density of a linear AR model can adequately describe it, and if not, what might be correct number of regimes. Multiple modes in the marginal distribution can be accounted for by accommodating each one of them with a regime in the GSMAR model. Skewness and many other forms of non-Gaussianity can also be accommodated with a mixture of normal or t-distributions, but it is less straightforward to determine the correct number of regimes. One should, nevertheless, be conservative with the choice is M, because with too many regimes in the model, some of the parameters are not identified (see Kalliovirta *et al.* 2015, Sections 3.1 and 3.2.2 and the references therein).

A kernel density estimate of the interest rate spread is depicted in the right panel of Figure 2 (black solid line). There are two visible modes in the density function, so a linear model (with unimodal error distribution) is clearly inadequate to describe it, while a two-regime mixture model could be appropriate. Even a three-regime model could be considered in order to explain the hump shape in the right tail of the distribution.

Examining the sample partial autocorrelation function (PACF) of the series can help in selecting the correct autoregresive order p, as for a pth order AR process, there should be a visible break in the PACF after the lag p. If the series is not autocorrelated, the sample partial autocorrelation function of the squared series may similarly help to detect the order

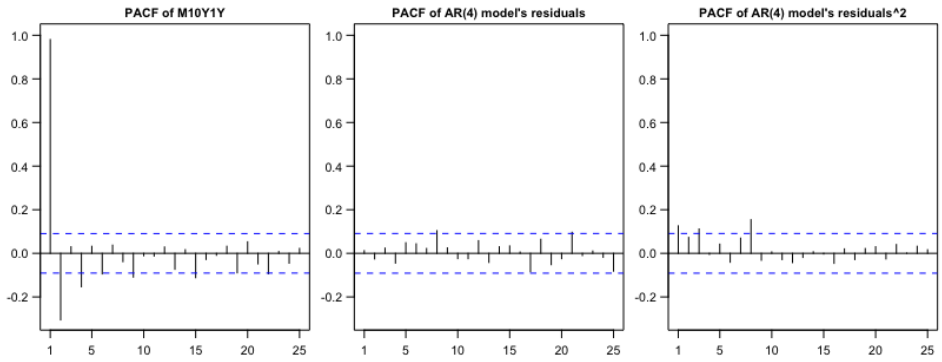

Figure 1: The sample partial autocorrelation function of the series `M10Y1Y` for the lags $1, ..., 25$ (on the right). The sample partial autocorrelation function of the Pearson residuals of a Gaussian AR(4) model (on the middle) and of the squared residuals (on the right) for the lags $1, ..., 25$. The blue dashed lines are the 95% critical bounds for partial autocorrelation coefficients of an IID process.

of ARCH type conditional heteroskedasticity. In the case of an autocorrelated series, it might be useful to first fit an AR model with a suitable autoregressive order, and then examine the PACF of the squared residuals. The sample partial autocorrelation function of the series `M10Y1Y` (calculated using the function `pacf` from the package **stats**, R Core Team 2021) is presented in the left panel of Figure 1.

Figure 1 shows that the PACF of the series has very large partial autocorrelation coefficient (PACC) at the first lag, relatively large PACCs at the second and fourth lags, and visibly smaller PACCs after the fourth lag. The autoregressive order $p = 4$ thereby seems like a reasonable candidate for a parsimonious AR model.[2] Hence, we fitted a Gaussian AR(4) model to the series and examined the PACF of its residuals and squared residuals, which are depicted in the middle and right panels of Figure 1, respectively.

The PACF of the AR(4) model's residuals shows that there is not much autocorrelation left in the residuals, so the autoregressive order $p = 4$ seems sufficient for capturing the autocorrelation structure of the series. The PACF of the AR(4) model's squared residuals shows PACCs slightly outside the 95% critical bounds at lags 2, 3, and 8. It thereby seems that the order 3 could be sufficient for modelling the (potentially present) ARCH type conditional heteroskedasticity, but larger order, such as 8, could be considered for a less parsimonious model. A StMAR model might, therefore, be appropriate with the autoregressive order $p = 4$. As discussed above, the two modes in the kernel density estimate of the series, on the other hand, indicate that two regimes seems like a good starting point for building the model.

If the candidate model is found inadequate, one may try to use a different autoregressive order p or to add a regime to the model (or switch to the StMAR model, if a GMAR model was found inadequate). Note that while with linear AR models increasing the autoregressive order typically improves the fitness, this is not necessarily the case with the GSMAR models, as the autoregressive order affects the regime-switching dynamics. In particular, because the mixing weights (6) are calculated using the whole joint distribution of the previous p observations, with a small p, the regime-switching probabilities react more sensitively to

[2]It turns out that the order $p = 4$ also minimizes the Akaike information criterion among the Gaussian AR(p) models, $p = 1, ..., 24$, based on the exact log-likelihood function (not shown).

individual observations than with a large p. It may hence be useful to also try to decrease the autoregressive order rather than just increase it.

In addition to comparing model adequacy (or forecasting accuracy, for example), information criteria can be utilized in the selection of the GSMAR model. **uGMAR** calculates the Akaike (AIC), Hannan-Quinn (HQIC), and Schwarz-Bayesian (BIC) information criteria. The values of the information criteria are not directly comparable for models with different autoregressive orders if the estimation is based on the conditional log-likelihood function, as the numbers of observations used in the estimation are different due to the different number of initial values. With the conditional log-likelihood function, the values of the information criteria can be divided by the number of observations used in the estimation (that is, the length of the series minus p) to obtain more comparable statistics. However, as the conditional estimation with each order p is based on slightly different observations, the comparison should be done with caution. The exact log-likelihood function, nonetheless, employs the full series in the estimation and thereby yields comparable values of information criteria for models with different orders p.

3.4. 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 2021b, Section 4).

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`). Some of the estimation rounds may end up in local-only maximum points or saddle points, but 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.

Section 3.3 concluded that a StMAR model with autoregressive order $p = 4$ and $M = 2$ mixture components seems like a reasonable candidate for modeling the monthly interest rate spread `M10Y1Y`. The following code fits this model to the series using the conditional log-likelihood function and performing 12 estimation rounds with eight 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 = 8, seeds = 4:15)

Using 8 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=11s
Results from the genetic algorithm:
The lowest loglik:  143.403
The mean loglik:    159.237
The largest loglik: 172.344
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=02s
Results from the variable metric algorithm:
The lowest loglik:  167.572
The mean loglik:    179.117
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 (Virolainen 2021b, Section 4). Specifically, overly large degrees of freedom parameter estimates may induce a nearly numerically singular Hessian matrix of the log-likelihood function when evaluated at the estimate, making the approximate standard errors and Kalliovirta's (2012) quantile residual tests often unavailable.

The estimates can be 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: 9.75

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

Regime 2
Mix weight: 0.19
Reg mean: 0.55
Var param: 0.01
Df param: 9348.94

y = [0.04] + [1.34]y.1 + [-0.59]y.2 + [0.54]y.3 + [-0.36]y.4 + [sigma_mt]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 unconditional mean μ_m, the variance parameter σ_m^2, and the degrees freedom parameter ν_m. For GMAR type regimes (if any), $\sigma_{m,t} = \sigma_m$ so the estimate of the variance parameter σ_m^2 is reported directly in the autoregressive formula.

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 2021b, Section 4). Switching to the appropriate G-StMAR model is recommended also because it removes the redundant degrees of freedom parameters from the model, thereby reducing its complexity. 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 estimate 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
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
Reg mean: 1.87
Var param: 0.04 (0.01)
Df param: 9.75 (4.17)
Reg var:  1.01

y = [0.06] + [1.28]y.1 + [-0.36]y.2 + [0.20]y.3 + [-0.15]y.4 + [sigma_mt]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. The mixing weight parameter estimate 0.19 of the GMAR type regime indicates that in the long run, roughly 19% of the observations are generated from this regime. Estimates of the unconditional mean and variance (0.55 and 0.14, respectively) are visibly smaller in the GMAR type regime than in the StMAR type regime (1.87 and 1.01, respectively). Hence, the GMAR type seems to mostly account for the periods when the series takes smaller values and is less volatile, while the StMAR type regime covers the more volatile periods of larger values. Interestingly, the AR parameters are somewhat similar in both regimes, implying that it could be appropriate to restrict them to be identical (this will be tested in Section 3.7).

Approximate standard errors are given in parentheses under or next to the related estimates. Note that the last mixing weight parameter estimate does not have an approximate standard error because it is not parametrized. Likewise, there is no standard error for the intercepts if mean parametrization is used (by setting `parametrization = "mean"` in `fitGSMAR`) and

vice versa. In order to obtain standard errors for the regimewise unconditional means or intercepts, one can easily swap between the mean and intercept parametrizations with the function `swap_parametrization`.

Missing values are reported when **uGMAR** is not able to calculate the standard error. This typically happens either because there is an overly large degrees of freedom parameter estimate in the model (as discussed above) or because the estimation algorithm did not stop a local maximum. In the latter case, the observed information matrix is not necessarily positive definite, implying that the diagonal entries of its inverse might not all be positive. Consequently, when extracting the approximate standard errors by taking the square roots of the diagonal entries from the inverse of the observed information matrix, the possibly present negative entries will lead to missing values.

Section 3.5 discusses how to evaluate with **uGMAR** whether the estimate is a local maximum (and how to improve the reliability of it being the global maximum). If the estimate is not a local maximum, one may try running more iterations of the variable metric algorithm with the function `iterate_more`. However, often when the algorithm does not stop a local maximum, it stopped to an unreasonable point very near the boundary of parameter space. As will be discussed next, in such a case it might be more appropriate to consider an alternative estimate that is clearly in the interior of the parameter space.

Other statistics reported in the summary printout include the log-likelihood and values of the 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 the 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 = 16, ncores = 8, seeds = 72:87)

Using 8 cores for 16 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=12s
Results from the genetic algorithm:
The lowest loglik:  140.441
The mean loglik:    155.421
The largest loglik: 167.858
Optimizing with a variable metric algorithm...
```

```
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=02s
Results from the variable metric algorithm:
The lowest loglik:  152.034
The mean loglik:    174.794
The largest loglik: 192.43
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 that the estimate might be inappropriate. 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: 192.43, AIC: -356.86, HQIC: -334.05, BIC: -298.90

Regime 1 (GMAR type)
Moduli of AR poly roots: 1.00, 1.00, 1.00, 1.00
Mix weight: 0.02 (0.03)
Reg mean: 2.65
Reg var:  0.13

y = [3.77] + [1.19]y.1 + [-1.81]y.2 + [1.19]y.3 + [-1.00]y.4 + sqrt[0.00]eps
    (0.02)   (0.01)       (0.01)      (0.01)       (0.00)          (0.00)

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.04, 1.93, 1.93, 1.48
Mix weight: 0.98
Reg mean: 0.89
Var param: 0.04 (0.01)
Df param: 4.98 (1.67)
Reg var:  1.75

y = [0.02] + [1.30]y.1 + [-0.36]y.2 + [0.21]y.3 + [-0.17]y.4 + [sigma_mt]eps
    (0.01)   (0.05)       (0.08)      (0.08)       (0.05)

Process mean: 0.92
Process var:  1.78
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 procedure. 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
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
Reg mean: 1.87
Var param: 0.04 (0.01)
Df param: 9.75 (4.14)
Reg var:  1.01

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

3.5. 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 2:

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

As Figure 2 (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. The hump shape in the right tail of the kernel density estimate is not explained by the mixture of the two distributions, but a third regime could be added for the purpose (for brevity, we do not study the three regime model further).

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 2021b, 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 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

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

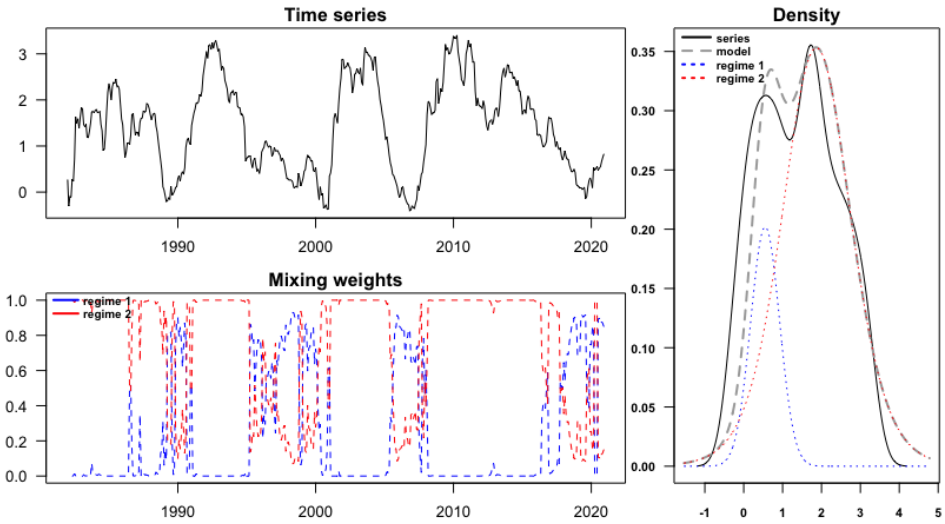

Figure 2: The figure produced by the command `plot(fit42gs)`. On the top left, the monthly spread between the 10-year and 1-year Treasury constant maturity rates, covering the period from 1982 January to 2020 December. On the bottom left, the estimated mixing weights of the G-StMAR model (`fit42gs`) 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).

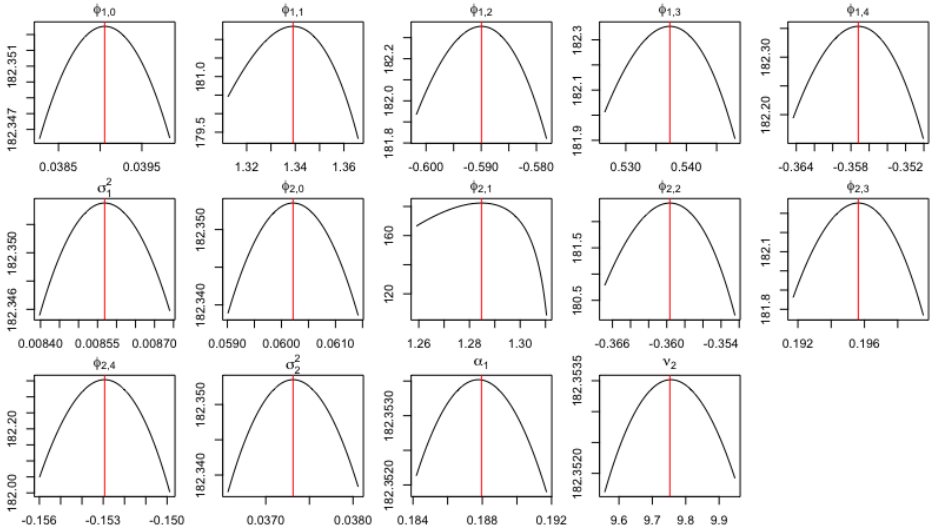

Figure 3: The figure produced by the command `profile_logliks(fit42gs)`. Graphs of the profile log-likelihood functions of the estimated G-StMAR model `fit42gs` with the red vertical lines pointing the estimates.

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]  0.0576396128 -0.0364233988 -0.0242331476 -0.0144442609 -0.0161249574
 [6]  0.0411603528 -0.0171471584 -0.0490156277 -0.0659635759 -0.0587742714
[11] -0.0635655297  0.0686981920 -0.0374653647  0.0002778317
```

and the following code calculates the second order condition:

```
R> get_soc(fit42gs)

 [1] -5.753554e-02 -1.354508e+01 -4.394382e+01 -6.467642e+01 -1.204519e+02
 [6] -1.672692e+02 -2.619181e+02 -8.869383e+02 -2.045380e+03 -4.862797e+03
[11] -4.355348e+04 -5.455077e+04 -2.727695e+05 -5.564824e+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 3, showing that the estimate's accuracy is reasonable, as changing any individual parameter value 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 number, 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 that have extremely spiky profile log-likelihood functions for some parameters and are thus difficult to find (see Appendix A). Therefore, it seems plausible that `fitGSMAR` also finds a reasonable ML estimate with a good reliability.

3.6. 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, the G-StMAR, $p = 4$, $M_1 = 1$, $M_2 = 1$ model (`fit42gs`) estimated in Section 3.4 obtained somewhat similar estimates for the AR parameters in both regimes. The following code estimates a version of this model such that the AR parameters are restricted to be the same in both regimes. Note that this model still allows for shifts in the conditional (and unconditional) mean, as the intercept parameters can vary across the regimes. The argument `print_res = FALSE` tells `fitGSMAR` not to the print the spread of the log-likelihoods obtained from each phase of estimation.

```
R> fit42gsr <- fitGSMAR(M10Y1Y, p = 4, M = c(1, 1), model = "G-StMAR",
+    restricted = TRUE, ncalls = 12, ncores = 8, seeds = 1:12,
+    print_res = FALSE)

Using 8 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=07s
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=01s
Finished!
```

The summary printout of the model shows the AR parameter estimates are the same in both regimes:

```
R> summary(fit42gsr)

Model:
 G-StMAR, p = 4, M1 = 1, M2 = 1, #parameters = 10, #observations = 468,
 conditional, intercept parametrization, AR parameters restricted, no
 constraints.
```

```
 log-likelihood: 180.02, AIC: -340.04, HQIC: -323.74, BIC: -298.64

Regime 1 (GMAR type)
Moduli of AR poly roots: 1.21, 1.83, 1.83, 1.21
Mix weight: 0.51 (0.17)
Reg mean: 2.13
Reg var:  0.46

y = [0.13] + [1.29]y.1 + [-0.40]y.2 + [0.25]y.3 + [-0.20]y.4 + sqrt[0.03]eps
    (0.03)   (0.05)       (0.08)      (0.08)       (0.05)          (0.00)

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.21, 1.83, 1.83, 1.21
Mix weight: 0.49
Reg mean: 0.54
Var param: 0.05 (0.06)
Df param: 2.76 (1.18)
Reg var:  0.83

y = [0.03] + [1.29]y.1 + [-0.40]y.2 + [0.25]y.3 + [-0.20]y.4 + [sigma_mt]eps
    (0.01)   (0.05)       (0.08)      (0.08)       (0.05)

Process mean: 1.35
Process var:  1.27
First p autocors: 0.98 0.95 0.91 0.87
```

In constrast to the unrestricted model, this model has larger regimewise unconditonal mean in the GMAR type regime than in the StMAR type regime. According to the unconditional regimewise variances, the StMAR type regime is the more volatilite regime in this model as well.

Whether imposing the constraints is reasonable, can be evaluated by employing a statistical test, comparing values of the information criteria, or examining the model adequacy, for example. As the summary printout shows, the information criteria values all decreased as opposed to the unrestricted model, implying that the constraints could be appropriate. Discussion on testing the constraints is postponed to Section 3.7, whereas diagnostics checks for evaluating the model adequacy are covered in Section 4.

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

$$\boldsymbol{\varphi}_m = \boldsymbol{C}_m\boldsymbol{\psi}_m, \quad m = 1, ..., M, \tag{12}$$

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

To exemplify, consider a GMAR model with autoregressive order $p = 3$ and $M = 2$ mixture components. To constrain the third AR coefficient of the second regime ($\varphi_{2,3}$) to zero but leaving the first regime unconstrained, 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",
+    constraints = C_list, ncalls = 12, ncores = 8, seeds = 1:12,
+    print_res = FALSE)

Using 8 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=05s
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=01s
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,
 conditional, intercept parametrization, not restricted, linear constraints
 imposed.

Regime 1
Mix weight: 0.56
Reg mean: 1.26

y = [0.02] + [1.25]y.1 + [-0.19]y.2 + [-0.07]y.3 + sqrt[0.01]eps

Regime 2
Mix weight: 0.44
Reg mean: 1.72

y = [0.07] + [1.27]y.1 + [-0.32]y.2 + [0.00]y.3 + sqrt[0.05]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.

3.7. Testing parameter constraints

One way to asses the validity of the imposed constraints is to compare the values of information criteria of the constrained and unconstrained models. **uGMAR**, however, also provides functions for testing the constraints with the likelihood ratio test and Wald test, which are applicable as the ML estimator of a GSMAR model has the conventional asymptotic distribution (as long as the model is correctly specified and one is willing to assume the validity of the required unverified assumptions, see Kalliovirta *et al.* 2015, pp. 254-255, Meitz *et al.* 2021, Theorem 3, and Virolainen 2021b, Theorem 2). For a discussion on the likelihood ratio and Wald tests, see Buse (1982) and the references therein, for example.

The likelihood ratio test considers the null hypothesis that the true parameter value $\boldsymbol{\theta}_0$ satisfies some constraints imposed on these parameters (such that the constrained parameter space is a subset of the parameter space, which is presented in Virolainen 2021b, Section 2.2 for the GSMAR models). Denoting by $\hat{L}_U$ and $\hat{L}_C$ the (maximized) log-likelihoods based on the unconstrained and constrained ML estimates, respectively, the test statistic takes the form

$$LR = 2(\hat{L}_U - \hat{L}_C). \tag{13}$$

Under the null, the test statistic is asymptotically χ^2-distributed with the degrees of freedom given by the difference in the dimensions of the unconstrained and constrained parameter spaces.

With **uGMAR**, the likelihood ratio test can be calculated with the function `LR_test`, which takes the unconstrained model (a class `gsmar` object) as its first argument and the constrained model as the second argument. For instance, in Section 3.6 we estimated a G-StMAR, $p = 4$, $M_1 = 1$, $M_2 = 1$ model such that the AR parameters are restricted to be equal in both regimes (the model `fit42gsr`), i.e., $\boldsymbol{\varphi}_1 = \boldsymbol{\varphi}_2$. The following code tests those constraints against the unconstrained model `fit42gs` with the likelihood ratio test and prints the results.

```
R> LR_test(fit42gs, fit42gsr)

        Likelihood ratio test

data:  fit42gs and fit42gsr
LR = 4.6695, df = 4, p-value = 0.3229
```

```
alternative hypothesis: the true parameter does not satisfy the constraints
imposed in fit42gsr
```

The large p-value indicates that we cannot reject the constraints at any conventional level of significance, and it might thereby be reasonable to consider the constrained model if it is found adequate.

uGMAR implements the Wald test of the null hypothesis

$$A\boldsymbol{\theta}_0 = c, \tag{14}$$

where A is a $(k \times d)$ matrix with full row rank, c is a $(k \times 1)$ vector, $\boldsymbol{\theta}_0$ is the true parameter value, d is the dimension of the parameter space, and k is the number of constraints. The Wald test statistic takes the form

$$W = (A\hat{\boldsymbol{\theta}} - c)'[A\mathcal{J}(\hat{\boldsymbol{\theta}})^{-1}A']^{-1}(A\hat{\boldsymbol{\theta}} - c), \tag{15}$$

where $\mathcal{J}(\hat{\boldsymbol{\theta}})$ is the observed information matrix evaluated at the ML estimate $\hat{\boldsymbol{\theta}}$. Under the null, the test statistic is asymptotically χ^2-distributed with k degrees of freedom (which is the difference in dimensions of the constrained and unconstrained parameter spaces).

With **uGMAR**, the Wald test can be calculated with function `Wald_test`, which takes the estimated unconstrained model (as a class `gsmar` object) as the first argument, the matrix A as the second argument, and the vector c as the third argument. To exemplify, we test whether the AR parameters and intercepts are identical in both regimes of the G-StMAR, $p = 4$, $M_1 = 1$, $M_2 = 1$ model, i.e., the null hypothesis $(\varphi_{1,0}, \boldsymbol{\varphi}_1) = (\varphi_{2,0}, \boldsymbol{\varphi}_2)$. The $(d \times 1)$ parameter vector $\boldsymbol{\theta}$ (which is presented at the end of Section 2.2 and again in Section 5) contains the intercept and AR parameters of the first regime in the entries $1, ..., 5$ and the intercept and AR parameters of the second regime in the entries $7, ..., 11$. The appropriate matrix A and vector c that state the hypothesis are set in the first two lines of the following code, and the third line calculates the test.

```
R> c <- rep(0, times = 5)
R> A <- cbind(diag(5), c, -diag(5), c, c, c)
Wald_test(fit42gs, A = A, c = c)

        Wald test

data:  fit42gs, A, c
W = 15.107, df = 5, p-value = 0.009916
alternative hypothesis: the true parameter theta does not satisfy
A%*%theta = c
```

As the above printout shows, the p-value is small enough to reject the null at the 1% level of significance, even though the identicality of the AR parameters was accepted with a large p-value by the likelihood ratio test. If one test identicality of the intercepts conditional on the constraint that the AR parameters are identical in both regimes (using the model `fit42gsr`), the Wald test produces the p-value 0.00025 (not shown for brevity). Thus, the intercepts

are not likely equal if the AR parameters are identical in both regimes.[3] As is demonstrated above, the Wald test has the benefit that it does not require estimation of the constrained model, and it is, therefore, not limited to the type of constraints **uGMAR** accommodates. The likelihood ratio test, on the other hand, is more conveniently calculated once the constrained model has been estimated.

Note that the standard tests are not applicable if the number of GMAR or StMAR type regimes is chosen too large, as then some of the parameters are not identified, causing the result of the asymptotic normality of the ML estimator to break down. This particularly happens when one tests for the number of regimes in the model, as the under the null some of the regimes are reduced from the model[4] (see the related discussion in Kalliovirta *et al.* 2015, Section 3.3.2). Similar caution applies for testing whether a regime is of the GMAR type against the alternative that it is of the StMAR type. Then $\nu_m = \infty$ under the null for the regime m to be tested, which violates the assumption that the parameter value is in the interior of a compact subset of the parameter space (see Virolainen 2021b, Theorem 2 and Assumption 1).

4. Quantile residual based model diagnostics

In the GSMAR models, the empirical counterparts of the error terms $\varepsilon_{m,t}$ in (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

$$R_t = \Phi^{-1}(F(y_t|\mathcal{F}_{t-1})), \quad t = 1, 2, ..., T, \tag{16}$$

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). Closed form expressions for the quantile residuals of the GSMAR processes are derived in Appendix B.

The empirical counterparts of the quantile residuals are calculated by using the parameter estimate and the observed data in (16). 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 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

[3] The test results do not, however, allow to infer that the process is likely bimodal, because GSMAR processes incorporating component processes with distinct means can have unimodal skewed marginal distributions. Moreover, one cannot infer about the (in)equality of the means of the component processes based on the (in)equality of the intercepts if the AR parameters are allowed vary freely. In particular, our null hypothesis $(\varphi_{1,0}, \boldsymbol{\varphi}_1) = (\varphi_{2,0}, \boldsymbol{\varphi}_2)$ does not test whether the component processes have identical means, as identical means can be obtained also with various other constraints. Identicality of the means can, however, be tested directly by switching to the mean parametrization (with the function `swap_parametrization`) and calculating the appropriate Wald test.

[4] Meitz and Saikkonen (2021) have, however, recently developed such tests for mixture models with Gaussian conditional densities.

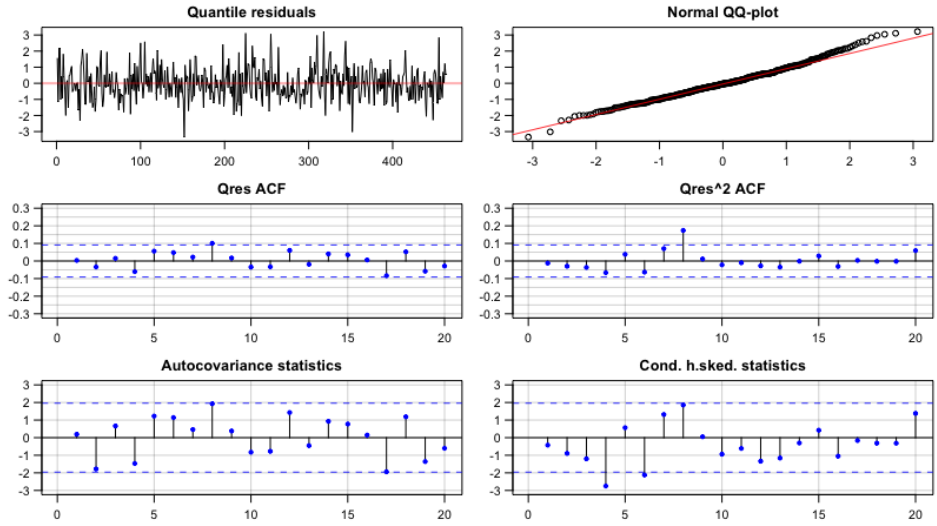

Figure 4: Diagnostic plot for the fitted model `fit42gsr` created using the function `diagnostic_plot`. The quantile residual time series (top left), normal quantile-quantile plot (top right), sample autocorrelation functions of the quantile residuals (middle left) and squared quantile residuals (middle 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 Kalliovirta's (2012) individual statistics they are the approximate 95% critical bounds.

`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 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 - particularly if the simulated sample is very large.

The likelihood ratio test accepted hypothesis that the AR coefficients of the G-StMAR $p = 4$, $M_1 = 1$, $M_2 = 2$ model are identical in both regimes (see Section 3.7). In order to evaluate whether this constrained model (`fit42gsr`) can adequately capture the autocorrelation structure, conditional heteroskedasticity, and distribution of the series, we create a diagnostic plot with the following code. We include Kalliovirta's (2012) individual statistic to the figure based on the observed data and calculated for the first 20 lags.

```
R> diagnostic_plot(fit42gsr, nlags = 20, plot_indstats = TRUE)
```

The resulting plot is presented in Figure 4. The quantile residual time series (the top left panel) has a period when it takes several consecutive negative values (roughly the observations $260, ..., 300$ with also some positive observations in between), but other than that it seems to

somewhat resemble an IID normal process. The normal quantile-quantile plot (the top right panel) shows that the quantile residuals' distribution has too fat right tail. This is possibly due to the inability to explain the hump shape in the right tail of the series' distribution with a mixture of one normal and one t-distribution, when the two modes are accounted for.

The sample autocorrelation function of the quantile residuals (the middle left panel) shows that there are no particularly large autocorrelation coefficients in the lags $1, ..., 20$. Moreover, as all Kalliovirta's (2012) autocorrelation statistics fall inside the asymptotic 95% critical bounds, the model seems to adequately describe the autocorrelation structure of the series. The sample autocorrelation function of the squared quantile residuals (the middle right panel), on the other hand, has a relatively large coefficient at the lag eight. Kalliovirta's (2012) conditional heteroskedasticity statistics (the bottom right panel) fall outside the asymptotic 95% critical bounds at the lags four and six, but at the lag eight the statistic is inside the bounds. Overall, it appears that in addition to the distribution, the model might not adequately explain the conditional heteroskedasticity of the series.

In order to employ the simulation procedure for calculating the individual statistics, 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).

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 restricted G-StMAR model `fit42gsr` 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> qrtr <- quantile_residual_tests(fit42gsr, lags_ac = c(1, 3, 6, 12),
+    nsimu = 10000)

Normality test p-value: 0.018

Autocorrelation tests:
lags | p-value
```

```
  1  |  0.849
  3  |  0.084
  6  |  0.488
  12 |  0.213

Conditional heteroskedasticity tests:
lags | p-value
  1  |  0.713
  3  |  0.299
  6  |  0.017
  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 12 lags. Also, the normality test and the heteroskedasticity test with six lags pass only at 1% level of significance. The rest of the tests, including all the autocorrelation tests pass at 5% level of significance, confirming our findings from examining the diagnostic plot: the model seem to adequately explain the autocorrelation structure of the series but struggles in capturing the distribution and conditional heteroskedasticity. Nevertheless, the inadequacies do not seem very serious.

Because the restricted model was found somewhat inadequate, we run the quantile residual tests for the unrestricted model as well in order to evaluate whether it captures the statistical properties of the series more adequately. The following code runs the same diagnostics tests for the unrestricted model `fit42gs`.

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

Normality test p-value: 0.087

Autocorrelation tests:
lags | p-value
  1  |  0.475
  3  |  0.020
  6  |  0.289
  12 |  0.077

Conditional heteroskedasticity tests:
lags | p-value
  1  |  0.579
  3  |  0.137
  6  |  0.002
  12 |  0.000
```

As the p-values show, relaxing the restrictions improved the model's capability to capture the distribution of the series but according to the test results, the unrestricted model does not explain conditional heteroskedasticity as well as the restricted one when taking into account

six lags (since the test now rejects at 1% level of significance). Also the autocorrelation test with three lags only passes at 1% level of significance. It thereby appears that the parsimonious restricted model could be more appropriate. Adding a third regime to the model or trying a different autoregressive order could also be considered for potentially improving the adequacy.

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 2021b, Section 4, and Section 3.4 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 3.4 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

$$\boldsymbol{\vartheta}_m=(\varphi_{m,0},\varphi_{m,1},...,\varphi_{m,p},\sigma_m^2),\quad m=1,...,M,\text{ and} \tag{17}$$

$$\boldsymbol{\nu}=(\nu_{M_1+1},...,\nu_M). \tag{18}$$

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. For brevity, they are only presented in the package documentation, 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 3.4 and 3.6. 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 implements the S3 method `simulate` for simulating observations from GSMAR processes. The method requires the process to be given as a class `gsmar` object, which are typically 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 `nsim` sets the length of the sample path to be simulated.

To give an example, the following code sets the random number generator seed to one and 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> mysim <- simulate(mod22, nsim = 500, seed = 1)
```

Our implementation of `simulate` 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 prediction intervals analytically for the GSMAR models is very complicated, so **uGMAR** employs the following simulation-based

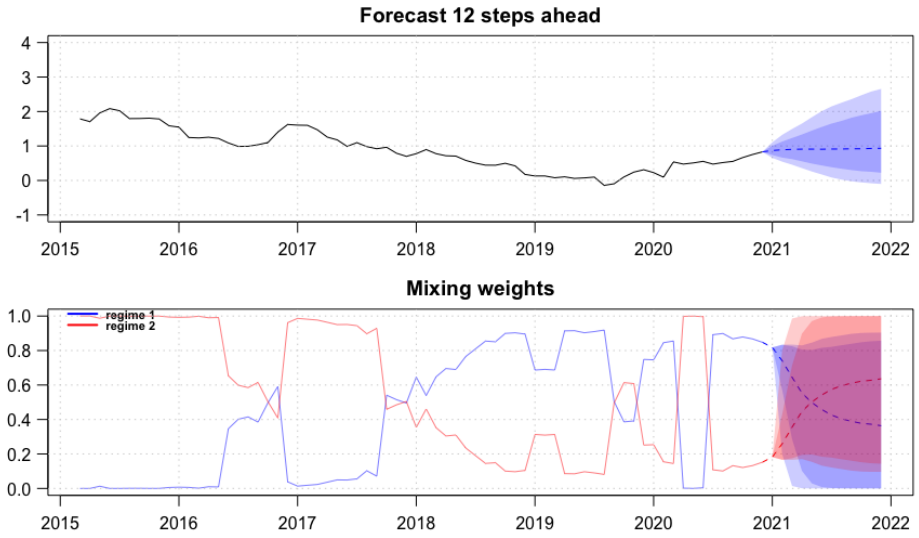

Figure 5: The figure created by the `predict` method for the G-StMAR model `fit42gs`. Twelve-months-ahead point prediction for the monthly interest rate spread (top) and the model's mixing weights (bottom) together with several preceding observations and prediction intervals with confidence levels 0.95 (outer interval) and 0.80 (inner interval).

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 unrestricted G-StMAR $p = 4, M_1 = 1, M_2 = 1$ model fitted to the monthly interest rate spread in Section 3.4 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 1.00  1.11
2   0.55 0.66   0.89 1.13  1.32
3   0.46 0.62   0.90 1.23  1.49
4   0.36 0.56   0.91 1.33  1.65
5   0.26 0.49   0.91 1.45  1.83
6   0.17 0.44   0.91 1.55  2.01
7   0.09 0.38   0.91 1.65  2.15
8   0.02 0.34   0.91 1.73  2.26
9  -0.02 0.30   0.92 1.82  2.36
10 -0.05 0.27   0.92 1.89  2.47
11 -0.08 0.25   0.93 1.95  2.58
12 -0.10 0.23   0.93 2.02  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 `plot_res = FALSE` in the arguments. The results can be plotted afterwards by using the `plot` method for the class `gsmarpred` objects that the `predict` method returns.

The figure created by the above example is presented in Figure 5. The point forecast does not predict any significant movements for the spread, but the prediction intervals appear to be skewed to the right. A possible explanation to the skewed prediction intervals is that at time of forecasting, the spread takes a value that is closer to the mean of the low-mean first regime than to the mean of the high-mean second regime. Hence, even if the process proceeds in the first regime, it does not (on average) move much lower, but switching to the second regime would (on average) lead to notably larger observations. Also, the forecast for the mixing weights reveals that after a few months, the high-mean second regime is predicted to become more probable than than the low-mean first regime, thus, explaining the skewed prediction intervals.

7. Summary

Mixture autoregressive models are useful for analyzing time series that exhibit nonlinear, 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.

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

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 a 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	`simulate` (method)	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 between mean and intercept parametrizations

Table 1: Some useful functions in **uGMAR** sorted according to their usage. The note "method" in parentheses 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`).

likelihood estimate is occasionally found very close to the boundary of the stationarity region of some regimes. We explained in Appendix A 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.

Computational details

The results in this paper were obtained using R 4.1.2 and **uGMAR** 3.4.1 package running on MacBook Pro 14", 2021, with Apple M1 Pro processor, 16 Gt of unified RAM, and macOS Monterey 12.1 operating system.

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 2006) is utilized to calculate some of the quantile residuals (16) 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 (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.

Some of the estimation results (and thereby everything that is calculated based on the estimates) may vary slightly when running the code on different computers. This is due to a small numerical error in the gradient of the log-likelihood function caused by the limited precision of the floating-point representation. The negligible numerical error accumulates in each iteration of the variable metric algorithm, which hence advances in slightly different paths on different computers (with given initial values). After a large number of iterations, the algorithm might therefore end up in slightly different points. This particularly occurs when there are StMAR type regimes in the model, possibly because there are many different pairs of degrees of freedom and variance parameter values that are relatively close to each other and yield almost the same log-likelihoods.

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

Aomoto K, Kita M (2011). *Theory of Hypergeometric Functions*. 1st edition. Springer-Verlag, Tokyo. doi:10.1007/978-4-431-53938-4.

Bollerslev T (1986). "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics*, **31**(3), 307–327. doi:10.1016/0304-4076(86)90063-1.

Boshnakov GN, Ravagli D (2021). ***mixAR**: Mixture Autoregressive Models*. R package version 0.22.5, URL https://CRAN.R-project.org/package=mixAR.

Buse A (1982). "The Likelihood Ratio, Wald, and Lagrange Multiplier Tests: An Expository Note." *The American Statistician*, **36**(3a), 153–157. doi:10.1080/00031305.1982.10482817.

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.

Engle RF (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." *Econometrica*, **50**(4), 987–1007. doi:10.2307/1912773.

Fong P, Li W, Yau C, Wong C (2007). "On a mixture vector autoregressive model." *The Canadian Journal of Statistics*, **35**(1), 135–150. doi:10.1002/cjs.5550350112.

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.

Ghalanos A (2020). ***rugarch**: Univariate GARCH Models*. R package version 1.4-4.

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

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

Hankin R (2007). "Very Large Numbers in R: Introducing Package **Brobdingnag**." *R News*, **7**(3).

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2021). ***forecast**: Forecasting Functions for Time Series and Linear Models*. R package version 8.15, URL https://pkg.robjhyndman.com/forecast/.

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 (2021). "A Mixture Autoregressive Model Based on Student's t-Distribution." *Communications in Statistics - Theory and Methods*. doi:10.1080/03610926.2021.1916531.

Meitz M, Saikkonen P (2021). "Testing for Observation-Dependent Regime Switching in Mixture Autoregressive Models." *Journal of Econometrics*, **222**(1), 601–624. doi:10.1016/j.jeconom.2020.04.048.

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 (2021). *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 (2018). ***gmvarkit**: Estimate Gaussian and Student's t Mixture Vector Autoregressive Models*. R package version 2.0.2, URL https://CRAN.R-project.org/package=gmvarkit.

Virolainen S (2021a). "Gaussian and Student's t mixture vector autoregressive model." *Unpublished working paper, available as arXiv:2109.13648*. URL https://arxiv.org/abs/2109.13648.

Virolainen S (2021b). "A Mixture Autoregressive Model Based on Gaussian and Student's t-distributions." *Studies in Nonlinear Dynamics & Econometrics*. doi:10.1515/snde-2020-0060.

Virolainen S (2021c). "Structural Gaussian Mixture Vector Autoregressive Model With Application to the asymmetric effects of monetary policy shocks." *Unpublished working paper, available as arXiv:2007.04713*. URL https://arxiv.org/abs/2007.04713.

Wong C, Li W (2000). "On a Mixture Autoregressive Model." *Journal of the Royal Statistical Society B*, **62**(1), 95–115. doi:10.1111/1467-9868.00222.

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.

Wuertz D, Setz T, Chalabi Y, Boudt C, Chausse P, Miklovac M (2020). ***fGarch**: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*. R package version 3042.83.2, URL https://CRAN.R-project.org/package=fGarch.

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 = 8, 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 1.000011). Also the variance parameter σ_2^2 is close to its lower bound zero (it is approximately $9 \cdot 10^{-6}$).

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 6 (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 6 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
$\hat{\boldsymbol{\theta}}_1$	0.58	0.56	0.10	0.61	7.85	−1.67	−1.00	0.00	0.99	1.42	6.86	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.

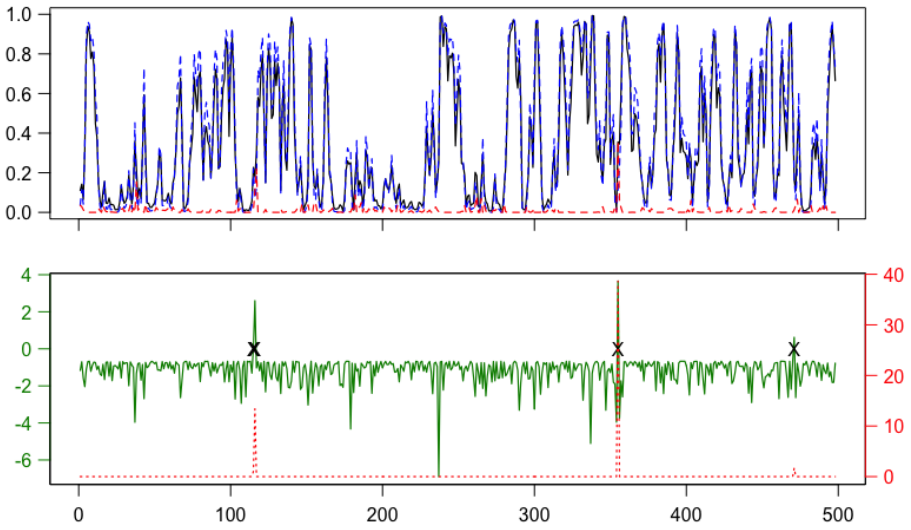

Figure 6: On the top, the GMAR $p = 2$, $M = 2$ process's second regime's true mixing weights (black solid line), the mixing weights based on the estimate $\hat{\boldsymbol{\theta}}_1$ in the second row of Table 2 (red dashed line), and the mixing weights based on the estimate $\hat{\boldsymbol{\theta}}_2$ 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 6 (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, observe 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.

B. Closed form expressions of quantile residuals

This section derives closed form expressions for the quantile residuals utilized by **uGMAR** and discussed in Section 4. For the GSMAR models, the quantile residuals are defined as

$$R_t = \Phi^{-1}(F(y_t|\mathcal{F}_{t-1})), \quad t = 1, 2, ..., T, \tag{19}$$

where $\Phi^{-1}(\cdot)$ is the standard normal quantile function,

$$F(y_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M} \alpha_{m,t} \int_{-\infty}^{y_t} f_m(u_t|\mathcal{F}_{t-1}) du_t \tag{20}$$

is the conditional cumulative distribution function of the considered GSMAR process (conditional on the previous observations), and $f_m(\cdot|\mathcal{F}_{t-1})$ is the conditional density function of the mth component process. To find a closed form expression for the quantile residuals defined in (19) and (20), it therefore suffices to solve the integrals $\int_{-\infty}^{y_t} f_m(u_t|\mathcal{F}_{t-1})du_t$, $m = 1, ..., M$, for GMAR type and StMAR type mixture components.

In the case of a GMAR type component, the conditional density function is the Gaussian density function with mean $\mu_{m,t}$ and variance σ_m^2. For $m \leq M_1$ in (20), we therefore have

$$\int_{-\infty}^{y_t} f_m(u_t|\mathcal{F}_{t-1})du_t = \int_{-\infty}^{y_t} n_1(u_t; \mu_{m,t}, \sigma_m^2)du_t = \Phi\left(\frac{u_t - \mu_{m,t}}{\sigma_m}\right), \tag{21}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

In the case of a StMAR type component, the conditional density function is the Student's t density function with mean $\mu_{m,t}$, variance $\sigma^2_{m,t}$, and ν_m+p degrees of freedom given as (Meitz *et al.* 2021, Appendix A)

$$t_1(u_t;\mu_{m,t},\sigma^2_{m,t},\nu_m+p)=\frac{\Gamma\left(\frac{1+\nu_m+p}{2}\right)}{\sqrt{\pi(\nu_m+p-2)}\Gamma\left(\frac{\nu_m+p}{2}\right)}\sigma^{-1}_{m,t}\left(1+\frac{(u_t-\mu_{m,t})^2}{(\nu_m+p-2)\sigma^2_{m,t}}\right)^{-(1+\nu_m+p)/2} \tag{22}$$

where $\Gamma(\cdot)$ is the gamma function. Taking use of the symmetry of the Student's t distribution about its mean $\mu_{m,t}$, we obtain

$$\int_{-\infty}^{y_t} f_m(u_t|\mathcal{F}_{t-1})du_t=\frac{1}{2}+\int_{\mu_{m,t}}^{y_t} t_1(u_t;\mu_{m,t},\sigma^2_{m,t},\nu_m+p)du_t. \tag{23}$$

By applying the change of variables $\tilde{u}_{m,t}\equiv u_t-\mu_{m,t}$ in the integral, the RHS of (23) can be expressed as

$$\frac{1}{2}+\frac{\Gamma\left(\frac{1+\nu_m+p}{2}\right)}{\sqrt{\pi(\nu_m+p-2)}\Gamma\left(\frac{\nu_m+p}{2}\right)}\sigma^{-1}_{m,t}\int_0^{\tilde{y}_{m,t}}\left(1+\frac{\tilde{u}^2_{m,t}}{a_{m,t}}\right)^{-b_m}d\tilde{u}_{m,t}, \tag{24}$$

where $\tilde{y}_{m,t}\equiv y_t-\mu_{m,t}$, $a_{m,t}\equiv(\nu_m+p-2)\sigma^2_{m,t}$, and $b_m\equiv(1+\nu_m+p)/2$. Then, by applying the change of variables $z_{m,t}\equiv\tilde{u}^2_{m,t}/\tilde{y}_{m,t}$, we can express the integral in the expression (24) as

$$\int_0^{\tilde{y}_{m,t}}\left(1+\frac{\tilde{u}^2_{m,t}}{a_{m,t}}\right)^{-b_m}d\tilde{u}_{m,t}=\frac{1}{2}\int_0^{\tilde{y}_{m,t}}\left(\frac{\tilde{y}_{m,t}}{z_{m,t}}\right)^{1/2}\left(1+\frac{z_{m,t}\tilde{y}_{m,t}}{a_{m,t}}\right)^{-b_m}dz_{m,t}. \tag{25}$$

By applying the third change of variables $x_{m,t}\equiv z_{m,t}/\tilde{y}_{m,t}$ and using the properties of the gamma function, the RHS of (25) can be expressed using a hypergeometric function as

$$\frac{\tilde{y}_{m,t}}{2}\int_0^1 x^{-1/2}_{m,t}\left(1-x_{m,t}\left(-\frac{\tilde{y}^2_{m,t}}{a_{m,t}}\right)\right)^{-b_m}dx_{m,t}=\tilde{y}_{m,t}\times{}_2\mathrm{F}_1\left(\frac{1}{2},b_m,\frac{3}{2};-\frac{\tilde{y}^2_{m,t}}{a_{m,t}}\right), \tag{26}$$

where the hypergeometric function is defined as (Aomoto and Kita 2011, Section 1.3.1)

$${}_2\mathrm{F}_1(a,b,c;x)=\frac{\Gamma(c)}{\Gamma(a)\Gamma(c-a)}\int_0^1 s^{a-1}(1-s)^{c-a-1}(1-sx)^{-b}ds, \tag{27}$$

when $|x|<1$, $a>0$, and $c-a>0$ (when $a,c\in\mathbb{R}$).

Using the above result, we have

$$\int_{-\infty}^{y_t} f_m(u_t|\mathcal{F}_{t-1})=\frac{1}{2}+\frac{\Gamma\left(\frac{1+\nu_m+p}{2}\right)}{\sqrt{\pi(\nu_m+p-2)}\Gamma\left(\frac{\nu_m+p}{2}\right)}\sigma^{-1}_{m,t}\tilde{y}_{m,t}\times{}_2\mathrm{F}_1\left(\frac{1}{2},b_m,\frac{3}{2};-\frac{\tilde{y}^2_{m,t}}{a_{m,t}}\right) \tag{28}$$

for $m>M_1$, whenever $\left|-\frac{\tilde{y}^2_{m,t}}{a_{m,t}}\right|<1$. That is, the closed form expression (28) exists when

$$|y_t-\mu_{m,t}|<\sqrt{(\nu_m+p-2)\sigma^2_{m,t}}. \tag{29}$$

If this condition does not hold, **uGMAR** calculates the quantile residual by numerically integrating the conditional density function $f_m(\cdot|\mathcal{F}_{t-1})$.

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