gmvarkit: A Family of Mixture Vector Autoregressive Models in R

Savi Virolainen
University of Helsinki

Abstract

We describe the R package **gmvarkit**, which provides tools for estimating and analyzing the reduced form and structural Gaussian mixture vector autoregressive model, the Student's *t* mixture vector autoregressive model, and the Gaussian and Student's *t* mixture vector autoregressive model. These three models constitute an appealing family of mixture autoregressive models that we call the GSMVAR 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, **gmvarkit** utilizes so-called quantile residuals and provides functions for graphical diagnostics as well as for calculating formal diagnostic tests. **gmvarkit** also enables to simulate from the GSMVAR processes, to forecast future values of the process using a simulation-based Monte Carlo method, and to estimate generalized impulse response functions as well as generalized forecast error variance decompositions. We illustrate the use of **gmvarkit** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator. This brief manuscript is currently intended as a vignette only but it was created with the Journal of Statistical Software style.

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

1. Introduction

Linear vector autoregressive (VAR) model is a standard tool in time series econometrics. It can be employed to answer questions about the statistical relationships of different variables or to forecast future values of the process, for example. Structural VAR model allows to trace out the effects of economic shocks that have been identified by the researcher. With an appropriate choice of the autoregressive order p, a linear VAR is often able to filter out autocorrelation from the series very well. If the errors are assumed to follow an autoregressive conditional heteroskedasticity (ARCH) process, the model is often able to also filter out conditional heteroskedasticity from the series.

In some cases, linear VAR models are not, however, not capable to capture all the relevant characteristics of the series. This includes shifts in the mean or volatility, and changes in the autoregressive 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, Kalliovirta, Meitz, and Saikkonen (2015), Kalliovirta *et al.* (2015), Meitz, Preve, and Saikkonen (2021), Virolainen (2020), and Virolainen (2021b,a). Following the recent developments by Kalliovirta, Meitz, and Saikkonen (2016), Virolainen (2020), and Virolainen (2021a), we consider the Gaussian mixture vector autoregressive (GMVAR) model, the Student's t mixture vector autoregressive (StMVAR) model, and the Gaussian and Student's t mixture autoregressive (G-StMVAR) model. These three models constitute an appealing family of mixture vector autoregressive models that we call the GSMVAR models.

A GSMVAR process generates each observation from one of its mixture components, which are either conditionally homoskedastic linear Gaussian vector autoregressions or conditionally heteroskedastic linear Student's t vector 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 **gmvarkit** providing a comprehensive set of easy-to-use tools for GSMVAR modeling, including unconstrained and constrained maximum likelihood (ML) estimation of the model parameters, quantile residual based model diagnostics, simulation from the processes, forecasting, and estimation generalized impulse response function (GIRF) as well as generalized forecast error variance decomposition (GFEVD). 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. However, 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, **gmvarkit** utilizes multivariate quantile residual diagnostics in the framework presented in Kalliovirta and Saikkonen (2010), including graphical diagnostics as well as diagnostic tests that take into account uncertainty about the true parameter value. Forecasting is based on a Monte Carlo simulation method. For univariate modeling, we suggest using the CRAN distributed R package **uGMAR** (Virolainen 2018).

The remainder of this paper is organized as follows...

Throughout this paper, we illustrate the use of **gmvarkit** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 to 2019Q4. 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 GMVAR model (Kalliovirta *et al.* 2016), the StMVAR model (Virolainen 2021a), and the G-StMVAR model (Virolainen 2021a), a family of mixture vector autoregressive models that we call the GSMVAR models. First, we define the component processes of the GSMVAR models - linear VARs based on either Gaussian or Student's t distribution. Then, we introduce the reduced form GSMVAR models. For brevity, we only give the definition of the more general G-StMVAR model but explain how the GMVAR and StMVAR 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. After defining the reduced form GSMVAR model, we introduce structural version of the models incorporating statistically identified structural shocks. Identification of the shocks is also briefly discussed.

3. Linear Gaussian and Student's t vector autoregressions

To develop theory and notation, consider first the component processes of the Gaussian and Student's t mixture vector autoregressive models. For a pth order linear Gaussian or Student's t vector autoregression z_t, we have

$$z_t = \phi_0 + \sum_{i=1}^{p} A_i z_{t-1} + \Omega_t^{1/2}\varepsilon_t, \quad \varepsilon \sim IID(0, I_d), \tag{1}$$

where $\Omega_t^{1/2}$ is a symmetric square root matrix of the positive definite $(d \times d)$ covariance matrix Ω_t, and $\phi_0 \in \mathbb{R}^d$. The $(d \times d)$ autoregression matrices are assumed to satisfy $\boldsymbol{A}_p \equiv [A_1 : ... : A_p] \in \mathbb{S}^{d \times dp}$, where

$$\mathbb{S}^{d \times dp} = \{[A_1 : ... : A_p] \in \mathbb{R}^{d \times dp} : \det(I_d - \sum_{i=1}^{p} A_i z^i) \neq 0 \text{ for } |z| \leq 1\} \tag{2}$$

defines the usual stability condition of a linear vector autoregression.

In the case of Gaussian VAR, the errors $\varepsilon_t areassumedstandardnormaldistributedandthecovariancematrices$ Ω are time invariant. Denoting $\boldsymbol{z}_t = (z_t, ..., z_{t-p+1})$ and $\boldsymbol{z}_t^+ = (z_t, \boldsymbol{z}_{t-1})$, it is well known that with IID Gaussian error process the stationary solution to (1) satisfies

$$\begin{aligned} \boldsymbol{z}_t &\sim n_{dp}(\mathbf{1}_p \otimes \mu, \Sigma_p) \\ \boldsymbol{z}_t^+ &\sim n_{d(p+1)}(\mathbf{1}_{p+1} \otimes \mu, \Sigma_{p+1}) \\ z_t|\boldsymbol{z}_{t-1} &\sim n_d(\phi_0 + \boldsymbol{A}_p \boldsymbol{z}_{t-1}, \Omega), \end{aligned} \tag{3}$$

where the last line defines the conditional distribution of z_t given $\boldsymbol{z}_{t-1}$. Denoting by $\Sigma(h)$ the lag h ($h = 0, \pm 1, \pm 2, ...$) autocovariance matrix of z_t, the quantities $\mu, \Sigma_p, \Sigma_1, \Sigma_{1p}, \Sigma_{p+1}$ are given as (see, e.g., Lütkepohl 2005, pp. 23, 28-29)

$$\begin{aligned}
\mu &= (I_d - \sum_{i=1}^{p} A_i)^{-1}\phi_0 && (d \times 1) \\
\text{vec}(\Sigma_p) &= (I_{(dp)^2} - \boldsymbol{A} \otimes \boldsymbol{A})^{-1}\text{vec}(\boldsymbol{\Omega}) && ((dp)^2 \times 1) \\
\Sigma_1 &= \Sigma(0) && (d \times d) \\
\Sigma(p) &= A_1\Sigma(p-1) + \cdots + A_p\Sigma(0) && (d \times d) \\
\Sigma_{1p} &= [\Sigma(1) : ... : \Sigma(p-1) : \Sigma(p)] = \boldsymbol{A}_p\Sigma_p && (d \times dp) \\
\Sigma_{p+1} &= \begin{bmatrix} \Sigma_1 & \Sigma_{1p} \\ \Sigma_{1p}' & \Sigma_p \end{bmatrix} && (d(p+1) \times d(p+1))
\end{aligned} \tag{4}$$

where

$$\Sigma_p = \underset{(dp \times dp)}{\begin{bmatrix} \Sigma(0) & \Sigma(1) & \cdots & \Sigma(p-1) \\ \Sigma(-1) & \Sigma(0) & \cdots & \Sigma(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma(-p+1) & \Sigma(-p+2) & \cdots & \Sigma(0) \end{bmatrix}},$$
$$\boldsymbol{A} = \underset{(dp \times dp)}{\begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I_d & 0 & \cdots & 0 & 0 \\ 0 & I_d & & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & I_d & 0 \end{bmatrix}}, \text{ and } \boldsymbol{\Omega} = \underset{(dp \times dp)}{\begin{bmatrix} \Omega & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}}. \tag{5}$$

Virolainen (2021a) shows that there are exists conditionally heteroskecastic Student's t vector autoregressions with distributional properties similar to the Gaussian VARs. Using the notation described above, these Student's t VARs are obtained by assuming $\varepsilon_t \sim t_d(0, I_d, \nu + dp)$ and

$$\Omega_t = \frac{\nu - 2 + (\boldsymbol{z} - \boldsymbol{1}_p \otimes \mu)'\Sigma_p^{-1}(\boldsymbol{z} - \boldsymbol{1}_p \otimes \mu)}{\nu - 2 + dp}\Omega. \tag{6}$$

These Student's t VARs are stationary and satisfy (Virolainen 2021a, Theorem 1)

$$\begin{aligned}
\boldsymbol{z}_t &\sim t_{dp}(\boldsymbol{1}_p \otimes \mu, \Sigma_p, \nu) \\
\boldsymbol{z}_t^+ &\sim t_{d(p+1)}(\boldsymbol{1}_{p+1} \otimes \mu, \Sigma_{p+1}, \nu) \\
z_t|\boldsymbol{z}_{t-1} &\sim t_d(\phi_0 + \boldsymbol{A}_p\boldsymbol{z}_{t-1}, \Omega_t, \nu + dp),
\end{aligned} \tag{7}$$

The conditional variance (6) consists of a constant covariance matrix that is multiplied by a time-varying scalar that depends on the quadratic form of the preceding p observations through the autoregressive parameters. In this sense, the model has a 'VAR(p)–ARCH(p)' representation, but the ARCH type conditional variance is not as general as in the conventional multivariate ARCH process (e.g., Lütkepohl 2005, Section 16.3) that allows the entries of the conditional covariance matrix to vary relative to each other.

We refer often to the linear Gaussian VARs as GMVAR type, because they are similar to the component processes of the GMVAR model (Kalliovirta *et al.* 2016). Likewise, we often refer to the linear Student's t VARs as StMVAR type, because they are similar to the component processes of the StMVAR model (Virolainen 2021a). The G-StMVAR model (Virolainen 2021a) incorporates both types of component processes. Because the GMVAR are StMVAR model are obtained as special cases of the G-StMVAR model by assuming that all the component processes are either GMVAR or StMVAR type, we will only give the definition of the more general G-StMVAR model.

4. Gaussian and Student's t mixture vector autoregressive model

Let y_t $(t = 1, 2, ...)$ be the real valued time series of interest, and let $\mathcal{F}_{t-1}$ denote σ-algebra generated by the random variables $\{y_s, s < t\}$. In a G-StMVAR model (Virolainen 2021a) with autoregressive order p and M mixture components (or regimes), the observations y_t are assumed to be generated by

$$y_t = \sum_{m=1}^{M} s_{m,t}(\mu_{m,t} + \Omega_{m,t}^{1/2}\varepsilon_{m,t}), \tag{8}$$

$$\mu_{m,t} = \phi_{m,0} + \sum_{i=1}^{p} A_{m,i}y_{t-i}, \tag{9}$$

where the following conditions hold.

Condition 1

1. *For $m = 1, ..., M_1 \leq M$, the random vectors $\varepsilon_{m,t}$ are IID $n_d(0, I_d)$ distributed, and for $m = M_1 + 1, ..., M$, they are IID $t_d(0, I_d, \nu_m + dp)$ distributed. For all m, $\varepsilon_{m,t}$ are independent of $\mathcal{F}_{t-1}$.*

2. *For each $m = 1, ..., M$, $\phi_{m,0} \in \mathbb{R}^d$, $\boldsymbol{A}_{m,p} \equiv [A_{m,1} : ... : A_{m,p}] \in \mathbb{S}^{d\times dp}$ (the set $\mathbb{S}^{d\times dp}$ is defined in (2)), and Ω_m is positive definite. For $m = 1, ..., M_1$, the conditional covariance matrices are constants, $\Omega_{m,t} = \Omega_m$. For $m = M_1 + 1, ..., M$, the conditional covariance matrices $\Omega_{m,t}$ are as in (6), except that $\boldsymbol{z}$ is replaced with $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$ and the regime specific parameters $\phi_{m,0}$, $\boldsymbol{A}_{m,p}$,Ω_m,ν_m are used to define the quantities therein. For $m = M_1 + 1, ..., M$, also $\nu_m > 2$.*

3. *The unobservable regime variables $s_{1,t}, ..., s_{M,t}$ are such that at each t, exactly one of them takes the value one and the others take the value zero according to the conditional probabilities expressed in terms of the ($\mathcal{F}_{t-1}$-measurable) mixing weights $\alpha_{m,t} \equiv Pr(s_{m,t} = 1|\mathcal{F}_{t-1})$ that satisfy $\sum_{m=1}^{M} \alpha_{m,t} = 1$.*

4. *Conditionally on $\mathcal{F}_{t-1}$, $(s_{1,t}, ..., s_{M,t})$ and $\varepsilon_{m,t}$ are assumed independent.*

The conditions $\nu_m > 2$ are made to ensure the existence of second moments. This definition implies that the G-StMVAR model generates each observation from one of its mixture components, linear Gaussian or Student's t vector autoregression discussed in Section **??**, and that the mixture component is selected randomly according to the probabilities given by the mixing weights $\alpha_{m,t}$. The first M_1 mixture components are assumed to be linear Gaussian VARs, and the last $M_2 \equiv M - M_1$ mixture components are assumed to be linear Student's t VARs. If all the component processes are Gaussian VARs ($M_1 = M$), the G-StMVAR model reduces to the GMVAR model of Kalliovirta *et al.* (2016). If all the component processes are Student's t VARs ($M_1 = 0$), we refer to the model as the StMVAR model.

The definition (8), (9), and Condition 1 leads to a model in which the conditional density function of y_t conditional on its past, $\mathcal{F}_{t-1}$, is given as

$$f(y_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M_1} \alpha_{m,t} n_d(y_t; \mu_{m,t}, \Omega_m) + \sum_{m=M_1+1}^{M} \alpha_{m,t} t_d(y_t; \mu_{m,t}, \Omega_{m,t}, \nu_m + dp). \tag{10}$$

The conditional densities $n_d(y_t; \mu_{m,t}, \Omega_{m,t})$ are obtained from (3), whereas $t_d(y_t; \mu_{m,t}, \Omega_{m,t}, \nu_m + dp)$ are obtained from Virolainen (2021a, Theorem 1). The explicit expressions of the density functions are given in an Appendix in Virolainen (2021a). To fully define the G-StMVAR model, it is then left to specify the mixing weights $\alpha_{m,t}$.

The mixing weights are defined as as weighted ratios of the component process stationary densities corresponding to the previous p observations. In order to formally specify the mixing weights, we first define the following function for notational convenience. Let

$$d_{m,dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m) = \begin{cases} n_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}), & \text{when } m \leq M_1, \\ t_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m), & \text{when } m > M_1, \end{cases} \tag{11}$$

where the dp-dimensional densities $n_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p})$ and $t_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m)$ correspond to the stationary distribution of the mth component process (given in equations (3) and (7)). Denoting $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$, the mixing weights of the G-StMVAR model are defined as

$$\alpha_{m,t} = \frac{\alpha_m d_{m,dp}(\boldsymbol{y}_{t-1}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m)}{\sum_{n=1}^{M} \alpha_n d_{n,dp}(\boldsymbol{y}_{t-1}; \mathbf{1}_p \otimes \mu_n, \Sigma_{n,p}, \nu_n)}, \tag{12}$$

where $\alpha_m \in (0, 1)$, $m = 1, ..., M$, are mixing weights parameters assumed to satisfy $\sum_{m=1}^{M} \alpha_m = 1$, $\mu_m = (I_d - \sum_{i=1}^{p} A_{m,i})^{-1} \phi_{m,0}$, and covariance matrix $\Sigma_{m,p}$ is given in (4) and (5) but using the regime specific parameters to define the quantities therein.

Because the mixing weights are weighted component process's stationary densities corresponding to the previous p observations, an observation is more likely to be generated from a regime with higher relative weighted likelihood. This is a convenient feature for forecasting but it also allows the researcher to associate specific characteristics to different regimes. Moreover, it turns out that this specific formulation of the mixing weights leads to attractive properties such as full knowledge of the stationary distribution of $p + 1$ consecutive observations and ergodicity of the process. Specifically, $\boldsymbol{y}_t = (y_t, ..., y_{t-p+1})$ has stationary distribution that is characterized by the density (Virolainen 2021a, Theorem 2)

$$f(\boldsymbol{y}) = \sum_{m=1}^{M} \alpha_m n_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}) + \sum_{m=M_1+1}^{M} \alpha_m t_{dp}(\boldsymbol{y}; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m). \tag{13}$$

gmvarkit collects the parameters of a G-StMVAR model to the $((M(d+d^2p+d(d+1)/2+2)-M_1-1)\times 1)$ vector $\boldsymbol{\theta} = (\boldsymbol{\vartheta}_1, ..., \boldsymbol{\vartheta}_M, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu})$, where $\boldsymbol{\vartheta}_m = (\phi_{m,0}, vec(\boldsymbol{A}_{m,p}), vech(\Omega_m))$ and $\boldsymbol{\nu} = (\nu_{M_1+1}, ..., \nu_M)$. The last mixing weight parameter α_M is not parametrized because it is obtained from the restriction $\sum_{m=1}^{M}\alpha_m = 1$. A G-StMVAR model with autoregressive order p, and M_1 GMVAR type and M_2 StMVAR type mixture components is referred to as G-StMVAR(p, M_1, M_2) model, whenever the order of the model needs to be emphasized. If the model imposes constraints, the parameter vector is different. For details, see the documentation.

5. Structural G-StMVAR model

We write the structural G-StMVAR model (Virolainen 2021a) as

$$y_t = \sum_{m=1}^{M} s_{m,t}(\phi_{m,0} + \sum_{i=1}^{p} A_{m,i}y_{t-i}) + B_t e_t \tag{14}$$

and

$$u_t \equiv B_t e_t = \begin{cases} u_{1,t} \sim n_d(0, \Omega_{1,t}) & \text{if} \quad s_{1,t} = 1 \quad \text{(with probability } \alpha_{1,t}) \\ \vdots & \\ u_{M_1,t} \sim n_d(0, \Omega_{M_1,t}) & \text{if} \quad s_{M_1,t} = 1 \quad \text{(with probability } \alpha_{M_1,t}) \\ u_{M_1+1,t} \sim t_d(0, \Omega_{M_1+1,t}, \nu_m + dp) & \text{if} \quad s_{M_1+1,t} = 1 \quad \text{(with probability } \alpha_{M_1,t}) \\ \vdots & \\ u_{M,t} \sim t_d(0, \Omega_{M,t}, \nu_m + dp) & \text{if} \quad s_{M,t} = 1 \quad \text{(with probability } \alpha_{M,t}) \end{cases} \tag{15}$$

where the probabilities are expressed conditionally on $\mathcal{F}_{t-1}$ and e_t $(d \times 1)$ in an orthogonal structural error. For the GMVAR type regimes, $m = 1, ..., M_1$, $\Omega_{m,t} = \Omega_m$. For the StMVAR type regimes, $m = M_1 + 1, ..., M$, $\Omega_{m,t} = \sigma_{m,t}^2\Omega_m$, where

$$\sigma_{m,t}^2 = \frac{\nu_m - 2 + (\boldsymbol{y}_{t-1} - \mathbf{1}_p \otimes \mu_m)'\Sigma_{m,p}^{-1}(\boldsymbol{y}_{t-1} - \mathbf{1}_p \otimes \mu_m)}{\nu_m - 2 + dp}. \tag{16}$$

The invertible $(d \times d)$ "B-matrix" B_t, which governs the contemporaneous relations of the shocks, is time-varying and a function of $y_{t-1}, ..., y_{t-p}$. With a particular choice of B_t, the conditional covariance matrix of the structural error can be normalized to an identity matrix. Consequently, a constant sized structural shock will be amplified according to the conditional variance of the reduced form error, thereby reflecting the specific state of the economy.

We have $\Omega_{u,t} \equiv \text{Cov}(u_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M_1}\alpha_{m,t}\Omega_m + \sum_{m=M_1+1}^{M}\alpha_{m,t}\sigma_{m,t}^2\Omega_m$, while the conditional covariance matrix of the structural error $e_t = B_t^{-1}u_t$ (which are not IID but are martingale differences and therefore uncorrelated) is obtained as

$$\text{Cov}(e_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M_1}\alpha_{m,t}B_t^{-1}\Omega_m B_t'^{-1} + \sum_{m=M_1+1}^{M}\alpha_{m,t}\sigma_{m,t}^2 B_t^{-1}\Omega_m B_t'^{-1}. \tag{17}$$

Therefore, we need to choose the B-matrix so that the structural shocks are orthogonal regardless of which regime they come from.

We employ the following decomposition to simultaneously diagonalize all the error term covariance matrices:

$$\Omega_m = W\Lambda_m W', \quad m = 1, ..., M, \tag{18}$$

where the diagonal of $\Lambda_m = \text{diag}(\lambda_{m1},...,\lambda_{md})$, $\lambda_{mi} > 0$ $(i = 1,...,d)$, contains the eigenvalues of the matrix $\Omega_m\Omega_1^{-1}$ and the columns of the nonsingular W are the related eigenvectors (that are the same for all m by construction). When $M = 2$, the decomposition (18) always exists, but for $M \geq 3$ its existence requires that the matrices share the common eigenvectors in W. This is, however, testable.

Lanne, Lütkepohl, and Maciejowsla (2010, Proposition 1) show that for a given ordering of the eigenvalues, W is unique apart from changing all signs a column, as long as for all $i \neq j \in \{1,...,d\}$ there exists an $m \in \{2,...,M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$ (for $m = 1$, $\Lambda_m = I_d$ and $\lambda_{m1} = \cdots = \lambda_{md} = 1$). A locally unique B-matrix that amplifies a constant sized structural shock according to the conditional variance of the reduced form error is therefore obtained as

$$B_t = W(\sum_{m=1}^{M_1}\alpha_{m,t}\Lambda_m + \sum_{m=M_1+1}^{M}\alpha_{m,t}\sigma_{m,t}^2\Lambda_m)^{1/2}. \tag{19}$$

Since $B_t^{-1}\Omega_m B_t'^{-1} = \Lambda_m(\sum_{n=1}^{M_1}\alpha_{n,t}\Lambda_n + \sum_{n=M_1+1}^{M}\alpha_{n,t}\sigma_{n,t}^2\Lambda_n)^{-1}$, the B-matrix (19) simultaneously diagonalizes $\Omega_1,...,\Omega_M$, and $\Omega_{u,t}$ (and thereby also $\Omega_{1,t},...,\Omega_{M,t}$) for each t so that $\text{Cov}(e_t|\mathcal{F}_{t-1}) = I_d$.

5.1. Identification of the structural shocks

With the decomposition (18) of $\Omega_1,...,\Omega_M$ and the B-matrix (19), a statistical identification of the shocks is achieved as long as each pair of the eigenvalues is distinct for some m. In order to identify structural shocks with economic interpretations, they need to be uniquely related to the economic shocks through the constraints on the B-matrix (or equally W) that only the shock of interest satisfies. Virolainen (2020, Proposition 1) gives formal conditions for global identification of any subset of the shocks when the relevant pairs eigenvalues are distinct in some regime. He also derives conditions for globally identifying some of the shocks when one of the relevant pairs of the eigenvalues is identical in all regimes. For convenience, we repeat the conditions in the former case below, but in the latter case, we refer to Virolainen (2020, Proposition 2).

Proposition 1 *Suppose $\Omega_m = W\Lambda_m W'$, $m = 1,...,M$, where the diagonal of $\Lambda = diag(\lambda_{m1},...,\lambda_{md})$, $\lambda_{mi} > 0$ $(i = 1,...,d)$, contains the eigenvalues of the matrix $\Omega_m\Omega_1^{-1}$ and the columns of the nonsingular W are the related eigenvectors. Then, the last d_1 structural shocks are uniquely identified if*

1. *for all $j > d - d_1$ and $i \neq j$ there exists an $m \in \{2,...,M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$,*
2. *the columns of W in a way that for all $i \neq j > d - d_1$, the ith column cannot satisfy the constraints of the jth column as is nor after changing all signs in the ith column, and*
3. *there is at least one (strict) sign constraint in each of the last d_1 columns of W.*

Condition 3 fixes the signs in the last d_1 columns of W, and therefore the signs of the instantaneous effects of the corresponding shocks. However, since changing the signs of the columns is effectively the same as changing the signs of the corresponding shocks, and the structural shock has a distribution that is symmetric about zero, this condition is not restrictive. The assumption that the last d_1 shocks are identified is not restrictive either, as one may always reorder the structural shocks accordingly.

For example, if $d = 3$, $\lambda_{m1} \neq \lambda_{m3}$ for some m, and $\lambda_{m2} \neq \lambda_{m3}$ for some m, the third structural shock can be identified with the following constraints:

$$B_t = \begin{bmatrix} * & * & * \\ + & + & - \\ + & + & + \end{bmatrix} \text{ or } \begin{bmatrix} - & * & + \\ - & + & - \\ * & + & + \end{bmatrix} \text{ or } \begin{bmatrix} + & 0 & - \\ * & * & * \\ + & * & + \end{bmatrix} \tag{20}$$

and so on, where "$*$" signifies that the element is not constrained, "$+$" denotes strict positive and "$-$" a strict negative sign constraint, and "0" means that the element is constrained to zero. Because imposing zero or sign constraints on W equals to placing them on B_t, they may be justified economically. Furthermore, besides a single sign constraint in each column, the constraints are over-identifying and can thus be also justified statistically. Sign constraints, however, don't reduce the dimension of the parameter space, making some of the measures such as the conventional likelihood ratio test and information criteria unsuitable for testing them. Quantile residual diagnostics, on the other hand, can be used to evaluate how well the restricted model is able to encapsulate the statistical properties of the data compared to the unrestricted alternative.

If condition 1 of Proposition 1 is strengthened to state that for all $i \neq j$ there exists an $m \in \{2, ..., M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$, the model is statistically identified even though only the last d_1 structural shocks have been identified with the proposition. Consequently, the constraints imposed in condition 2 become testable. If it cannot be assumed that all the pairs of the eigenvalues are distinct in some regime, then the testing problem is nonstandard and the conventional asymptotic distributions of likelihood ratio and Wald test statistics become unreliable. Note, however, that since placing zero or sign constraints on W equals to placing them on the B-matrix (19), the constraints imposed in condition 2 can be justified economically as usual.

6. Impulse response analysis

6.1. Generalized impulse response function

We consider the generalized impulse response function (GIRF) Koop, Pesaran, and Potter (1996) defined as

$$\text{GIRF}(n, \delta_j, \mathcal{F}_{t-1}) = \text{E}[y_{t+n}|\delta_j, \mathcal{F}_{t-1}] - \text{E}[y_{t+n}|\mathcal{F}_{t-1}], \tag{21}$$

where n is the chosen horizon, $\mathcal{F}_{t-1} = \sigma\{y_{t-j}, j > 0\}$ as before, the first term in the RHS is the expected realization of the process at time $t + n$ conditionally on a structural shock of magnitude $\delta_j \in \mathbb{R}$ in the jth element of e_t at time t and the previous observations, and the second term in the RHS is the expected realization of the process conditionally on the previous observations only. GIRF thus expresses the expected difference in the future outcomes when the specific structural shock hits the system at time t as opposed to all shocks being random.

Due to the p-step Markov property of the SG-StMVAR model, conditioning on (the σ-algebra generated by) the p previous observations $\boldsymbol{y}_{t-1} \equiv (y_{t-1}, ..., y_{t-p})$ is effectively the same as conditioning on $\mathcal{F}_{t-1}$ at the time t and later. The history $\boldsymbol{y}_{t-1}$ can be either fixed or random, but with random history the GIRF becomes a random vector, however. Using fixed $\boldsymbol{y}_{t-1}$ makes sense when one is interested in the effects of the shock in a particular point of time, whereas more general results are obtained by assuming that $\boldsymbol{y}_{t-1}$ follows the stationary distribution of the process. If one is, on the other hand, concerned about a specific regime, $\boldsymbol{y}_{t-1}$ can be assumed to follow the stationary distribution of the corresponding component model.

In practice, the GIRF and its distributional properties can be approximated with a Monte Carlo algorithm that generates independent realizations of the process and then takes the sample mean for point estimate. If $\boldsymbol{y}_{t-1}$ is random and follows the distribution G, the GIRF should be estimated for different values of $\boldsymbol{y}_{t-1}$ generated from G, and then the sample mean and sample quantiles can be taken to obtain the point estimate and confidence intervals. The algorithm implemented in ‘gmvarkit‘ is presented in an Appendix of Virolainen (2020).

Because the SG-StMVAR model allows to associate specific features or economic interpretations for different regimes, it might be interesting to also examine the effects of a structural shock to the mixing weights $\alpha_{m,t}$, $m = 1, ..., M$. We then consider the related GIRFs $E[\alpha_{m,t+n}|\delta_j, \boldsymbol{y}_{t-1}] - E[\alpha_{m,t+n}|\boldsymbol{y}_{t-1}]$ for which point estimates and confidence intervals can be constructed similarly to (21).

In **gmvarkit**, the GIRF can be estimated with the function `GIRF` which should be supplied with the estimated SG-StMVAR model or a SG-StMVAR model build with hand specified parameter values using the function `GMVAR`. The size of the structural shock can be set with the argument `shock_size`. If not specified, the size of one standard deviation is used; that is, the size one. Among other arguments, the function may also be supplied with the argument `init_regimes` which specifies from which regimes' stationary distributions the initial values are generated from. If more than one regime is specified, a mixture distribution with weights given by the mixing weight parameters is used. Alternatively, one may specify fixed initial values with the argument `init_values`. Note that the confidence intervals (whose confidence level can be specified with the argument `ci`) reflect uncertainty about the initial value only and not about the parameter estimates.

Because estimating the GIRF and confidence intervals for it is computationally demanding, parallel computing is taken use of to shorten the estimation time. The number of CPU cores used can be set with the argument `ncores`. The objects returned by the `GIRF` function have their own `plot` and `print` methods. Also, cumulative impulse responses of the specified variables can be obtained directly by specifying the argument `which_cumulative`, while scaled GIRFs can be obtained by specifying the argument `scale` as desired.

6.2. Generalized forecast error variance decomposition

We consider the generalized forecast error variance decomposition (GFEVD) **?** that is defined for variable i, shock j, and horizon n as

$$\text{GFEVD}(n, y_{it}, \delta_j, \mathcal{F}_{t-1}) = \frac{\sum_{l=0}^{n} \text{GIRF}(l, \delta_j, \mathcal{F}_{t-1})_i^2}{\sum_{k=1}^{d} \sum_{l=0}^{n} \text{GIRF}(l, \delta_k, \mathcal{F}_{t-1})_i^2}, \tag{22}$$

where n is the chosen and $\text{GIRF}(l, \delta_j, \mathcal{F}_{t-1})_i$ is the ith element of the related GIRF (see also the notation described for GIRF in the previous section). That is, the GFEVD is otherwise similar to the conventional forecast error variance decomposition but with GIRFs in the place of conventional impulse response functions. Because the GFEVDs sum to unity (for each variable), they can be interpreted in a similar manner to the conventional FEVD.

In **gmvarkit**, the GFEVD can be estimated with the function `GFEVD`. As with the GIRF, the GFEVD is dependent on the initial values. The type of the initial values is set with the argument `initval_type`, and there are three options:

1. `"data"` which estimates the GIRFs for all possible length p histories in the data and then the GIRFs in the GFEVD are obtained as the sample mean over those GIRFs.

2. `"random"` which generates the initial values from the stationary distribution of the process or from the mixture of the stationary distributions of some specific regime(s) with the relative mixing proportions given by the mixing weight parameters. The initial regimes can be set with the argument `init_regimes`. The GIRFs in the GFEVD are obtained as the sample mean over the GIRFs estimated for the different random initial values.

3. `"fixed"` which estimates the GIRFs for a single fixed initial value that is set with the argument `init_ values`.

The shock size is the same for all scalar components of the structural shock and it can be adjusted with the argument `shock_size`. If the GIRFs for some variables should be cumulative before calculating the GFEVD, specify them with the argument `which_cumulative`. Finally, note that the GFEVD objects have their own plot and print methods.

SIIRRÄ TÄMÄ KAPPALE MYÖHEMMÄKSI

7. Estimation and model selection

7.1. Log-likelihood function

gmvarkit employs the method of maximum likelihood (ML) for estimating the parameters of the G-StMVAR model. Even the exact log-likelihood function is available, as we have established the stationary distribution of the process in Theorem **??**. 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-StMVAR model takes the form

$$L(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M}\alpha_m d_{m,dp}(\boldsymbol{y}_0;\mathbf{1}_p\otimes\mu_m,\Sigma_{m,p},\nu_m)\right) + \sum_{m=1}^{M} l_t(\boldsymbol{\theta}), \tag{23}$$

where $d_{m,dp}(\cdot;\mathbf{1}_p\otimes\mu_m,\Sigma_{m,p},\nu_m)$ is defined in (11) and

$$l_t(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M_1}\alpha_{m,t}n_d(y_t;\mu_{m,t},\Omega_m) + \sum_{m=M_1+1}^{M}\alpha_{m,t}t_d(y_t;\mu_{m,t},\Omega_{m,t},\nu_m+dp)\right). \tag{24}$$

If stationarity of the initial values seems unreasonable, one can condition on the initial values and base the estimation on the conditional log-likelihood function, which is obtained by dropping the first term on the right hand side of (23).

Virolainen (2021a, Theorem 3) shows that the ML estimator of the G-StMVAR model is strongly consistent and has the conventional high-level conditions. In the case of a GMVAR model ($M_1 = M$), however, establishing asymptotic normality of the ML estimator requires less unverified assumptions (Kalliovirta *et al.* 2016, Theorem 3).

If there are two regimes in the model ($M = 2$), the structural G-StMVAR model is obtained from estimated reduced form model by decomposing the covariance matrices $\Omega_1, ..., \Omega_M$ as in (18). If $M \geq 3$ or overidentifying constraints are imposed on B_t through W, the model can be reparametrized with W and Λ_m ($m = 2, ..., M$) instead of $\Omega_1, ..., \Omega_M$, and the log-likelihood function can be maximized subject to the new set of parameters and constraints.[1] In this case, the decomposition (18) is plugged in to the log-likelihood function and $vech(\Omega_1), ..., vech(\Omega_M)$ are replaced with $vec(W)$ and $\boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_M$ in the parameter vector $\boldsymbol{\theta}$, where $\boldsymbol{\lambda}_m = (\lambda_{m1}, ..., \lambda_{md})$.

7.2. Two-phase estimation procedure

Finding the ML estimate amounts maximizing the log-likelihood function (23) (and (24)) over a high dimensional parameter space satisfying the constraints summarized in Assumption **??**. Due to the complexity of the log-likelihood function, numerical optimization methods are required. The maximization problem can, however, be challenging in practice. This is particularly due to the mixing weights' complex dependence on the preceding observations, which induces a large number of modes to the surface of the log-likelihood function, and large areas to the parameter space where it is flat in multiple directions. Also, the popular EM algorithm (Redner and Walker 1984) is virtually useless here, as at each maximization step one faces a new optimization problem that is not much simpler than the original one. Following Meitz, Preve, and Saikkonen (2018); Meitz *et al.* (2021) and Virolainen (2021b, 2018), we therefore employ a two-phase estimation procedure in which a genetic algorithm is used to find starting values for a gradient based method.

The genetic algorithm in **gmvarkit** 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), where the genetic algorithm is discussed in the univariate context. After running the genetic algorithm, the estimation is finalized with a variable metric algorithm (Nash 1990, algorithm 21, implemented by R Core Team 2020) using central difference approximation for the gradient of the log-likelihood function.

7.3. Examples of unconstrained estimation

In this section, we demonstrate how to estimate GSMVAR models with **gmvarkit** and provide several examples in order to illustrate various frequently occurring situations. In addition to the ordinary estimation, we particularly show how a GSMVAR model can be built based on a local-only maximum point when the ML estimate seems unreasonable. We also consider the estimation of the appropriate G-StMVAR model when the estimated StMVAR model contains overly large degrees of freedom estimates (see the related discussion in Virolainen

[1] Namely, instead of constraining $vech(\Omega_1), ..., vech(\Omega_M)$ so that $\Omega_1, ..., \Omega_M$ are positive definite, we impose the constraints $\lambda_{mi} > 0$ for all $m = 2, ..., M$ and $j = 1, ..., d$.

2021a). In the examples, we only consider $p = 1$ models for simplicity and because then the code outputs fit in the margins, estimation times are shorter, etc.

In **gmvarkit**, the GSMVAR models are defined as class `gsmvar` S3 objects, which can be created with given parameter values using the constructor function `GSMVAR` (see Section **??**) or by using the estimation function `fitGSMVAR`, which estimates the parameters and then builds the model. For estimation, `fitGSMVAR` 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 `"GMVAR"`, `"StMVAR"`, or `"G-StMVAR"`. For GMVAR and StMVAR models, the argument `M` is a positive integer, whereas for the G-StMVAR model it is a length two numeric vector specifying the number of GMVAR type regimes in the first element and the number of StMVAR type regimes in the second.

Additional arguments may be supplied to `fitGSMVAR` in order to specify, for example, whether the exact log-likelihood function should be used instead of the conditional one (`conditional`), 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. **In general, we recommend being conservative with choice of M due to the identifation problems induced if the number of regimes is chosen too large. Also, estimation of models that contain more than two regimes can be extremely challenging.**

We illustrate the use of **gmvarkit** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 to 2019Q4. The following code fits a StMVAR($p = 1, M = 2$) this model to this series (`gdpdef`) using the conditional log-likelihood function and performing two estimation rounds with two CPU cores. **In practice, hundreds or even thousands of estimation rounds is often required to obtain reliable results. The larger the dimension of the series is and the larger the order of the model is, the more estimation rounds is required.** We use only two estimation rounds in this simplistic example to shorten the estimation time, knowing beforehand that the given seeds produce the desired result (in this simplistic case, almost all the estimation rounds end up to the MLE, however).

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> data("gdpdef", package = "gmvarkit")
R> m1 <- fitGSMVAR(gdpdef, p=1, M=2, model="StMVAR", ncalls=2, seeds=1:2)

Using 2 cores for 2 estimations rounds...
Optimizing with a genetic algorithm...
Results from the genetic algorithm:
```

```
The lowest loglik:  -254.476
The mean loglik:    -254.15
The largest loglik: -253.823
Optimizing with a variable metric algorithm...
Results from the variable metric algorithm:
The lowest loglik:  -247.496
The mean loglik:    -245.24
The largest loglik: -242.985
Calculating approximate standard errors...
Finished!
```

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 because at least one the regimes contains a near-singular covariance matrix. This kind of unreasonable boundary points can often be disregarded, and the model can be built based on a reasonable estimate found from a local maximum that is clearly in the interior of the parameter space. Models based on the next-best local maximum can be built with the function `alt_gsmvar` by adjusting its argument `which_largest`.

The following code builds a StMVAR model based on the second-largest local maximum found in the estimation:

```
R> m2 <- alt_gsmvar(m1, which_largest=2)
```

The estimates can be examined with the `print`.

```
R> print(m2)
```

```
Reduced form StMVAR model:
 p = 1, M = 2, d = 2, #parameters = 21, #observations = 244 x 2,
 conditional log-likelihood, intercept parametrization, no AR parameter constraints

Regime 1
Mixing weight: 0.83
Regime means: 0.78, 0.54
Df parameter:  7.57

   Y     phi0          A1                           Omega
1 y1 = [ 0.55 ] + [  0.33 -0.04 ] y1.1 + (          [  0.42 0.00 ] )
2 y2   [ 0.12 ]   [  0.05  0.71 ] y2.1   ( ARCH_mt [  0.00 0.04 ] )
  1/2
1     eps1
2     eps2

Regime 2
Mixing weight: 0.17
```

```
Regime means: 0.66, 1.67
Df parameter:  58889.20

   Y     phi0         A1                            Omega
1 y1 = [ 1.60 ] + [  0.13 -0.61 ] y1.1 + (         [  1.21 -0.04 ] )
2 y2   [ 0.48 ]   [ -0.03  0.72 ] y2.1   ( ARCH_mt [ -0.04  0.14 ] )
  1/2
1     eps1
2     eps2
```

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

$$y_t = \varphi_{m,0} + \varphi_{m,1}y_{t-1} + \dots + \varphi_{m,p}y_{t-p} + \sigma_{m,t}\varepsilon_{m,t}. \tag{25}$$

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 2020, 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: 9.76 (4.25)
Reg var:  1.01

y = [0.06] + [1.28]y.1 + [-0.36]y.2 + [0.20]y.3 + [-0.15]y.4 + [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 7.6).

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 7.4 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 **??** 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 = 12, ncores = 4, seeds = 1:12)

Using 4 cores for 12 estimation rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=30s
Results from the genetic algorithm:
The lowest loglik:  133.926
The mean loglik:    147.787
The largest loglik: 175.58
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=10s
Results from the variable metric algorithm:
The lowest loglik:  151.91
The mean loglik:    172.875
The largest loglik: 185.558
Finished!
Warning message:
In warn_ar_roots(ret) :
  Regime 1 has near-unit-roots! Consider building a model from the next-
  largest local maximum with the function 'alt_gsmar' by adjusting its
  argument 'which_largest'.
```

The function throws a warning, because the largest found maximum point incorporates a regime that is very close to the boundary of the stationarity region, indicating 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: 185.56, AIC: -343.12, HQIC: -320.30, BIC: -285.16

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

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

Regime 2 (StMAR type)
Moduli of AR poly roots: 1.03, 1.92, 1.92, 1.54
Mix weight: 0.98
Reg mean: 0.80
Var param: 0.03 (0.01)
Df param: 5.94 (1.97)
Reg var:  1.65

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

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

The summary statistics reveal that there are four near-unit-roots in the GMAR type regime and the variance parameter estimate is very small. Such estimates often occur when there are several regimes in the model and the estimation algorithm is ran a large number of times.

If one suspects that the estimate is inappropriate, it is easy to build a model based on the second-largest maximum point that was found in the estimation 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: 9.76 (4.05)
Reg var:  1.01

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

7.4. Further examination of the estimates

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

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

As Figure 1 (the top and bottom left panels) shows, the first regime prevails when the spread takes small values, while the second regime mainly dominates when the spread takes large values. The graph of the model's marginal stationary density (the right panel), on the other hand, shows that the two regimes capture the two modes in the marginal distribution of the spread. 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 2020, 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

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} \approx \frac{f(\boldsymbol{\theta} + \boldsymbol{h}^{(i)}) - f(\boldsymbol{\theta} - \boldsymbol{h}^{(i)})}{2h}, \quad h > 0, \tag{26}$$

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 `figures/plotfit42gs.png`

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

`figures/proflogfit42gs.png`

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

the other hand, as the surface of the profile log-likelihood function is very flat about a large degrees of freedom parameter estimate, large differences work well for the approximation.

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

```
R> get_foc(fit42gs)
```

```
 [1]  0.1122667304  0.0443675437  0.0412785752  0.0423330799  0.0452292663
 [6]  0.3538279050 -0.0812537190 -0.0732948896 -0.0755336131 -0.0767990779
[11] -0.0825067588 -0.0117666801 -0.0004755852 -0.0003115161
```

and the following code calculates the second order condition:

```
R> get_soc(fit42gs)
```

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

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

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

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

The output is displayed in Figure 2, showing that the estimate's accuracy is reasonable, as changing any individual parameter 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 **??**). Therefore, it seems plausible that `fitGSMAR` also finds a reasonable ML estimate with a good reliability.

7.5. Examples of constrained estimation

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

7.6. 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 2020, Theorem 2). For a discussion on the likelihood ratio and Wald tests, see **?** 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 2020, 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

$$LR = 2(\hat{L}_U - \hat{L}_C). \tag{27}$$

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 7.5 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`). 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

$$A\boldsymbol{\theta}_0 = c, \tag{28}$$

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

$$W = (A\hat{\boldsymbol{\theta}} - c)^{\top}[A\mathcal{J}(\hat{\boldsymbol{\theta}})^{-1}A^{\top}]^{-1}(A\hat{\boldsymbol{\theta}} - c), \tag{29}$$

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 the same constraints as with the likelihood ratio test, i.e., that the AR parameters satisfy $\varphi_{1,i} = \varphi_{2,i}$ for $i = 1, ..., 4$, in the G-StMAR, $p = 4$, $M_1 = 1$, $M_2 = 1$ model. The $(d \times 1)$ parameter vector $\boldsymbol{\theta}$ (which is presented at the end of Section **??** and again in Section 9) contains the AR parameters of the first regime in the entries $2, ..., 5$ and the AR parameters of the second regime in the entries $8, ..., 11$. 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 = 4)
R> A <- cbind(c, diag(4), c, c, -diag(4), c, c, c)
R> Wald_test(fit42gs, A = A, c = c)


        Wald test

data:  fit42gs, A, c
W = 4.6272, df = 4, p-value = 0.3277
alternative hypothesis: the true parameter theta does not satisfy
A%*%theta = c
```

As the above printout shows, the p value is similar to the one obtained by the likelihood ratio test. The Wald test, however, has the benefit that is 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[2] (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 2020, Theorem 2 and Assumption 1).

8. Quantile residual based model diagnostics

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

$$R_t = \Phi^{-1}(F(y_t|\mathcal{F}_{t-1})), \quad t = 1, 2, ..., T, \tag{30}$$

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 presented in Appendix **??**.

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

9. 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

$$\boldsymbol{\vartheta}_m = (\varphi_{m,0}, \varphi_{m,1}, ..., \varphi_{m,p}, \sigma_m^2), \quad m = 1, ..., M, \text{ and} \tag{31}$$

$$\boldsymbol{\nu} = (\nu_{M_1+1}, ..., \nu_M). \tag{32}$$

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

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

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 7.3 and 7.5. If one wishes to parametrize the model with the regimewise unconditional means (μ_m) instead of the intercepts ($\varphi_{m,0}$), the argument `parametrization` should be set to `"mean"` in which case the intercept parameters $\varphi_{m,0}$ are replaced with μ_m in the parameter vector. By default, **uGMAR** uses intercept parametrization.

To exemplify, we build the GMAR $p = 2$, $M = 2$ model that is used in the simulation experiment in Appendix **??**. 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.

10. Simulation and forecasting

10.1. 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 **??** from the GMAR process built in Section 9:

```
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`.

10.2. 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 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"`.

11. Summary

FILL IN

Computational details

The results in this paper were obtained using R 4.0.2 and **uGMAR** 2.0.0 package running on MacBook Pro 13", 2020, with Intel Core i5-8257U 1.40Ghz processor and 8 Gt 2133 Mhz LPDDR3 RAM.

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`).

References

Dorsey R, Mayer W (1995). "Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features." *Journal of Business and Economic Statistics*, **13**(1), 53–66. doi:10.1080/07350015.1995.10524579.

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.

Kalliovirta L, Saikkonen P (2010). "Reliable Residuals for Multivariate Nonlinear Time Series Models." *Unpublished revision of HECER discussion paper No. 247*. URL https://blogs.helsinki.fi/lkvaisan/files/2010/08/ReliableResiduals.pdf.

Koop G, Pesaran M, Potter S (1996). "Impulse response analysis in nonlinear multivariate models." *Journal of Econometrics*, **74**(1), 119–147. doi:10.1016/0304-4076(95)01753-4.

Lanne M, Lütkepohl H, Maciejowsla K (2010). "Structural vector autoregressions with Markov switching." *Journal of Economic Dynamics and Control*, **34**(2), 121–131. doi:10.1016/j.jedc.2009.08.002.

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.

Lütkepohl H (2005). *New Introduction to Multiple Time Series Analysis*. 1st edition. Springer, Berlin. doi:10.1007/978-3-540-27752-1.

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.

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

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

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

Redner R, Walker H (1984). "Mixture Densities, Maximum Likelihood and the Em Algorithm." *Society for Industrial and Applied Mathematics*, **26**(2), 195–239. `doi:10.1137/1026034`.

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). *uGMAR: Estimate Univariate Gaussian or Student's t Mixture Autoregressive Model*. R package version 3.4.0 available at CRAN: `https://CRAN.R-project.org/package=uGMAR`, URL `https://CRAN.R-project.org/package=uGMAR`.

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

Virolainen S (2021a). "Gaussian and Student's t Mixture Vector Autoregressive Model." *Unpublished working paper, available as arXiv:2109.13648*. URL `https://arXiv: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`.

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`

`figures/diagplotfit42gsr.png`

Figure 3: 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.