sstvars: Structural Smooth Transition Vector Autoregressive Models in R

Savi Virolainen
University of Helsinki

Abstract

We describe the R package **sstvars**, which provides tools for estimating and analyzing the reduced form and structural smooth transition vector autoregressive (STVAR) models. The package implements various transition weight functions, conditional distributions, identification methods, and parameter restrictions. 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, **sstvars** utilizes residuals based diagnostics and provides functions for graphical diagnostics and for calculating formal diagnostic tests. **sstvars** also accommodates the estimation of linear impulse response functions, nonlinear generalized impulse response functions, and generalized forecast error variance decompositions. Further functionality includes hypothesis testing, plotting the profile log-likelihood functions about the estimate, simulation from STVAR processes, and forecasting, for example. We illustrate the use of **sstvars** 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.

Keywords: smooth transition vector autoregressive model, structural smooth transition vector autoregressive model, regime-switching, SVAR, STVAR, maximum likelihood estimation.

1. Introduction

Linear vector autoregressive (VAR) models are a standard tools in time series econometrics. They 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 models, in particular, allow to trace out the effects of economic shocks. With an appropriate choice of the autoregressive order p, a linear VAR model 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 also often able to adequately filter out conditional heteroskedasticity from the series.

In some cases, linear VAR models are not, however, capable of capturing all the relevant characteristics of the data. 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 this kind of regime-switching behav-

ior have been proposed, one of them is the smooth transition vector autoregressive (STVAR) models that allow to capture gradual shifts in the dynamics of the data. They consist of a finite number of regimes, each of which are linear vector autoregressions that are characterized by different autoregressive coefficients or error term covariance matrices. The package **sstvars** considers STVAR models in which, at each point of time, the observation is a weighted average of the conditional means of the regimes plus a random error whose covariance matrix is a weighted average of the covariance matrices of the regimes. The weights, in turn, are expressed in terms of time-varying transition weights that depend on the preceding observations. Different STVAR models can be created by specifying the transition weights or the error distribution in various ways.

This manuscript describes the R package **sstvars** providing a set of easy-to-use tools for STVAR modeling, including unconstrained and constrained maximum likelihood (ML) estimation of the model parameters, residual based model diagnostics, estimation of linear impulse response functions, nonlinear generalized impulse response functions, and generalized forecast error variance decompositions. Further functionality includes hypothesis testing, plotting the profile log-likelihood functions about the estimate, simulation from STVAR processes, and forecasting, for example. Various transition weight functions are accommodated, including logistic weights (Anderson and Vahid 1998), multinomial logit weights, exponential weights (e.g., Hubrich and Teräsvirta 2013), threshold weights (Tsay 1998), and transition weights that defined as weighted relative likelihoods of the regimes corresponding to the preceding p observations (FILL IN REFERENCE). Currently, the accommodated conditional distributions include Student's t distribution and Gaussian distribution, whereas the accommodated identification methods include recursive identification and identification by heteroskedasticity. More conditional distributions and identification methods are probably added in the future, however.

The estimation of the model parameters can, in some cases, be rather tricky. This is because the transition weights are determined endogenously, which induces a large number of modes to the log-likelihood function and large areas of the parameter space where the log-likelihood function is flat in multiple directions. Therefore, 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.

The remainder of this paper is organized as follows. Section 2 defines the implemented STVAR models and discusses some of their properties. Identification of the structural shocks is also discussed. Section 4 discusses estimation of the model parameters. We also illustrate how the STVAR models can be estimated and examined with **sstvars** and how various parameter restrictions can be imposed in the estimation. Section 5 discusses how to evaluate the model adequacy with **sstvars** using residual based diagnostics. Section 6 discusses impulse response analysis, including generalized impulse response functions and generalized forecast error variance decompositions. Section 7 shows how the STVAR models can be constructed with given parameter values. In Section 8, we first show how to simulate observations from a STVAR process, and then we illustrate how to forecast future values of a STVAR process with a simulation-based Monte Carlo method. Finally, Section 9 concludes and collects some useful

functions in **sstvars** to a single table for convenience. Throughout this paper, we illustrate the use of **sstvars** 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.

2. Smooth Transition Vector Autoregressive Models

This section describes the STVAR models implemented in **sstvars**. First, we describe the general framework of STVAR models accommodated by **sstvars** and present a sufficient condition for their ergodic stationarity. Then, we present the implemented specifications of transition weight functions and conditional distributions. Finally, we discuss structural STVAR models and implemented methods for identification of the shocks.

2.1. General framework for STVAR models

Let y_t, $t = 1, 2, ...$, be the d-dimensional time series of interest and $\mathcal{F}_{t-1}$ denote the σ-algebra generated by the random vectors $\{y_{t-j}, j > 0\}$. We consider STVAR models with M regimes and autoregressive order p assumed to satisfy

$$y_t = \sum_{m=1}^{M} \alpha_{m,t}\mu_{m,t} + B_t e_t, \quad e_t \sim IID(0, I_d) \tag{1}$$

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

$$B_t B_t' = \sum_{m=1}^{M} \alpha_{m,t}\Omega_m, \tag{3}$$

where $\phi_{m,0} \in \mathbb{R}^d$ are intercept parameters, $\Omega_1, ..., \Omega_M$ are the positive definite $(d \times d)$ covariance matrices of the regimes, B_t is an invertible $(d \times d)$ impact matrix that governs the contemporaneous relationships of the shocks and is a function of $\{y_{t-j}, j = 1, ..., p\}$. For the reduced form model, any invertible matrix B_t that satisfies Equation (3) can be assumed without loss of generality, whereas structural models assume a specific structure on B_t. The structural errors e_t $(d \times 1)$ are independent and identically distributed with mean zero and identity covariance matrix, and they are independent of $\mathcal{F}_{t-1}$. The transition weights $\alpha_{m,t}$ are assumed to be $\mathcal{F}_{t-1}$-measurable functions of $\{y_{t-j}, j = 1, ..., p\}$ and to satisfy $\sum_{m=1}^{M} \alpha_{m,t} = 1$ at all t. They express the proportions of the regimes the process is on at each point of time. Through, a STVAR model with autoregressive order p and M regimes is referred to as a STVAR(p, M) model, whenever the order of the model needs to be emphasized.

Conditional on $\mathcal{F}_{t-1}$, the conditional mean of the above described process is $\mu_{y,t} \equiv E[y_t|\mathcal{F}_{t-1}] = \sum_{m=1}^{M} \alpha_{m,t}\mu_{m,t}$, and the conditional covariance matrix is $\Omega_{y,t} \equiv \text{Cov}(y_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M} \alpha_{m,t}\Omega_m$. In other words, the conditional mean is a weighted sum the regime-specific means $\mu_{m,t}$ with the weights given by the transition weights $\alpha_{m,t}$, whereas the conditional covariance matrix is a weighted sum of the regime-specific conditional covariance matrices Ω_m. Different STVAR models are obtained by specifying the transition weights or the error distribution (i.e., the conditional distribution) in various ways. See Hubrich and Teräsvirta (2013) for a survey on STVAR literature, including formulations more general than our framework.

The package **sstvars** accommodates models in which the transition weights are functions of $\{y_{t-j}, j = 1, ..., p\}$, which is required for applicability of the stationarity condition presented below. Moreover, $\mathcal{F}_{t-1}$-measurability of the transition weights ensures that the true generalized impulse responses functions (Koop, Pesaran, and Potter 1996) can be easily estimated, as completely exogenous switching-variables are excluded from affecting the weights. We also assume that the transition weights are identical for all the individual equations in (1) and (3), which is also required for applicability of the stationarity condition. Consequently, at each t, the process can be described as a weighted sum of linear vector autoregressions.

Stationarity condition

It can be shown that a sufficient condition for the ergodic stationarity of the STVAR model (1)-(3) can expressed in terms of the joint spectral radius (JSR) of certain matrices (Kheifets and Saikkonen 2020). The JSR of a finite set of square matrices $\mathcal{A}$ is defined by

$$\rho(\mathcal{A}) = \limsup_{j\to\infty} \left(\sup_{A\in\mathcal{A}^j} \rho(A) \right)^{1/j}, \tag{4}$$

where $\mathcal{A}^j = \{A_1 A_2 ... A_j : A_i \in \mathcal{A}\}$ and $\rho(A)$ is the spectral radius of the square matrix A. Consider the companion form AR matrices of the regimes defined as

$$\boldsymbol{A}_m = \underset{(dp\times dp)}{\begin{bmatrix} A_{m,1} & A_{m,2} & \cdots & A_{m,p-1} & A_{m,p} \\ I_d & 0 & \cdots & 0 & 0 \\ 0 & I_d & & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & I_d & 0 \end{bmatrix}}, \; m = 1, ..., M. \tag{5}$$

Kheifets and Saikkonen (2020, Theorem 1) and FILL IN REFERENCE TO LANNE AND VIROLAINEN WORKING PAPER show that if the following condition holds, the STVAR process is ergodic stationary (both strictly and second-order).

Condition 1 $\rho(\{\boldsymbol{A}_1, ..., \boldsymbol{A}_M\}) < 1$.

Condition 1 is, however, computationally demanding the check in practice with a reasonable accuracy (e.g., Chang and Blondel 2013), making it impractical to use in the estimation. Therefore, we consider a necessary condition for Condition 1 that is easier to check in practice, which is that the usual stability condition is satisfied for each of the regimes. Specifically, the following condition, which is analogous to Corollary 1 of Kheifets and Saikkonen (2020), is necessary for Condition 1.

Condition 2 $\max\{\rho(\boldsymbol{A}_1), ..., \rho(\boldsymbol{A}_M)\} < 1$,

where $\rho(\boldsymbol{A}_m)$ is the spectral radius of $\boldsymbol{A}_m$, $m = 1, ..., M$.

Note that validity Condition 2 does not imply the validity of Condition 1 that guarantees stationarity of the model. However, in practice models that satisfy Condition 2 and are not very close to breaking this condition should most often satisfy Condition 1. For

checking the validity of Condition 1, **sstvars** implements the formula of Parrilo and Jadbabaie (2008) for the upper bound proposed by Blondel and Nesterov (2005) (the function `bound_joint_spectral_radius`). But unfortunately the calculations required are computationally very demanding in practice when the model is not small and even a relatively tight upper bound is required to verify Condition 1. Therefore, in the cases where Condition 1 needs to be formally verified, we suggest using the MATLAB toolbox **JSR** by Jungers (2023), which automatically combines several methods for bound the JSR and finds an accurate upper bound with a more reasonable computational effort.

2.2. Specifications of the conditional distribution

Currently, **sstvars** accommodates two types of error distributions: Gaussian and Student's t, which are discussed below.

Gaussian STVAR model

Assuming the structural errors e_t have standard normal distributions, the conditional distribution of y_t conditional on $\mathcal{F}_{t-1}$ is Gaussian and characterized by the density function

$$f(y_t|\mathcal{F}_{t-1}) = n_d(y_t; \mu_t, \Omega_t) = (2\pi)^{-d/2} \det(\Omega_t)^{-1/2} \exp\left\{-\frac{1}{2}(y_t - \mu_t)'\Omega_t^{-1}(y_t - \mu_t)\right\}. \quad (6)$$

That is, the conditional distribution is simply the d-dimensional Gaussian distribution with mean μ_t and covariance matrix Ω_t. The Gaussian distribution simple and can be used with all of our transition weight functions, but in some cases it is useful to employ the more heavy tailed Student's t distribution instead.

Student's t STVAR model

To accommodate more heavy tailed data, instead of using Gaussian errors one may consider Student's t errors and assume the shocks e_t are Student's t distributed with the mean zero, identity covariance matrix, and $\nu > 2$ degrees of freedom (where the assumption $\nu > 2$ is made to ensure the existence of second moments). The Student's t STVAR model has the conditional distribution, conditional $\mathcal{F}_{t-1}$, characterized by the density function

$$f(y_t|\mathcal{F}_{t-1}) = t_d(y_t; \mu_t, \Omega_t, \nu) = C_d(\nu)\det(\Omega_t)^{-1/2}\left(1 + \frac{(y_t - \mu_t)'\Omega_t^{-1}(y_t - \mu_t)}{\nu - 2}\right)^{-(d+\nu)/2}, \quad (7)$$

where

$$C_d(\nu) = \frac{\Gamma\left(\frac{d+\nu}{2}\right)}{\sqrt{\pi^d(\nu-2)^d}\Gamma\left(\frac{\nu}{2}\right)}, \quad (8)$$

and $\Gamma(\cdot)$ is the gamma function. The conditional distribution is, hence, the d-dimensional Student's t distribution with mean μ_t, covariance matrix Ω_t, and ν degrees of freedom. Note that the parametrization differs from the conventional one, as the distribution is parametrized with a covariance matrix instead a scale matrix (see, e.g., Meitz, Preve, and Saikkonen 2023, Appendix A for details about the parametrization).

The Student's t errors are more flexible than the Gaussian ones, but they cannot be used with the transition weight function that is defined as weighted ratios of the regime's stationary

densities (see Section 2.3.1). This is because it requires the knowledge of the stationary distributions of the regimes corresponding to p consecutive observations, and the stationary distribution is not known for the Student's t regimes. Moreover, Student's t STVAR models are more difficult estimate in practice than the Gaussian ones, and estimation of the STVAR models can be demanding.

2.3. Specifications of the transition weights

Various specifications of the transition weights $\alpha_{m,t}$ can be considered to obtain smooth transition VARs with different properties. We assume that the transition weights are functions of $\{y_{t-j}, j = 1, ..., p\}$, because it is required for applicability of the stationarity condition discussed in Section 2.1.1. Moreover, $\mathcal{F}_{t-1}$-measurability of the transition weights ensures that the true generalized impulse responses functions can be easily estimated, as completely exogenous switching-variables are excluded from affecting the weights.[1] We also assume that the transition weights are identical for all the individual equations in (1) and (3), which is required for applicability of the stationarity condition. Consequently, at each t, the process can be described as a weighted sum of linear VARs.

Weighted relative likelihoods

If the conditional distribution is specified to be Gaussian, weighted relative likelihoods of the regimes can be used as transition weights (FILL IN REFERENCE TO LANNE AND VIROLAINEN). In this specification, the transitions weights depend on the full distribution of the preceding p observations, they are defined identically to the mixing weights in the Gaussian mixture vector autoregressive (GMVAR) model of Kalliovirta, Meitz, and Saikkonen (2016). Denoting $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$, the transition weights are defined as

$$\alpha_{m,t} = \frac{\alpha_m n_{dp}(\boldsymbol{y}_{t-1}; \mathbf{1}_p \otimes \mu_m, \boldsymbol{\Sigma}_{m,p})}{\sum_{n=1}^{M} \alpha_n n_{dp}(\boldsymbol{y}_{t-1}; \mathbf{1}_p \otimes \mu_n, \boldsymbol{\Sigma}_{n,p})}, \quad m = 1, ..., M, \tag{9}$$

where $\alpha_1, ..., \alpha_M$ are transition weight parameters that satisfy $\sum_{m=1}^{M} \alpha_m = 1$ and $n_{dp}(\cdot; \mathbf{1}_p \otimes \mu_m, \boldsymbol{\Sigma}_{m,p})$ is the density function of the dp-dimensional Gaussian distribution with mean $\mathbf{1}_p \otimes \mu_m$ and covariance matrix $\boldsymbol{\Sigma}_{m,p}$. The symbol $\mathbf{1}_p$ denotes a p-dimensional vector of ones, $\otimes$ is Kronecker product, $\mu_m = (I_d - \sum_{i=1}^{p} A_{m,i})^{-1}\phi_{m,0}$, and the covariance matrix $\boldsymbol{\Sigma}_{m,p}$ is given in Lütkepohl (2005, Equation (2.1.39)), but using the parameters of the mth regime. That is, $n_{dp}(\cdot; \mathbf{1}_p \otimes \mu_m, \boldsymbol{\Sigma}_{m,p})$ corresponds to the density function of the stationary distribution of the mth regime.

The transition weights are thus weighted ratios of the stationary densities of the regimes corresponding to the preceding p observations. This specification is appealing, as it states that the greater the weighted relative likelihood of a regime is, the greater the weight of this regime is. The regimes are, hence, formed based on the statistical properties of the data and are not affected by the choice of the switching variables similarly to the logistic weights.

In the GMVAR model (Kalliovirta *et al.* 2016), the definition of the mixing weights also leads to attractive theoretical properties such as the the knowledge of the stationary distribution of $p + 1$ consecutive observations. But this is not the case in our STVAR model, as the

[1] Weaker forms of exogeneity of specific variables can, however, be imposed by constraining the AR matrices $A_{m,i}$, $m = 1, ..., M$, $i = 1, ..., p$, or the impact matrix B_t accordingly (see Section 4.6)

structure of the model is different. The GMVAR model has been implemented to the R package **gmvarkit** (Virolainen 2018a), which works quite similarly to **sstvars**.

Logistic transition weights

A common specification assumes logistic transition weights (e.g., Anderson and Vahid 1998) that vary according to the level of the switching variable, which we assume to be a lagged endogenous variable. Here we assume that the model has only two regimes ($M = 2$), and in the next section, we show how the logistic weights generalize to multinomial logit weights that accommodate more regimes.

The logistic transition weights are defined as

$$\alpha_{1,t} = 1 - \alpha_{2,t}, \tag{10}$$

$$\alpha_{2,t} = [1 + \exp\{-\gamma(y_{it-j} - c)\}]^{-1}, \tag{11}$$

where y_{it-j} is the jth lagged observation ($j \in \{1, ..., p\}$) of the ith variable ($i \in \{1, ..., d\}$), $c \in \mathbb{R}$ is a location parameter, and $\gamma > 0$ is a scale parameter. The location parameter c determines the mid point of the transition function, i.e., the value of the (lagged) switching variable when the weights are equal. The scale parameter γ, in turn, determines the smoothness of the transitions (smaller γ implies smoother transitions), and it is assumed strictly positive so that $\alpha_{2,t}$ is increasing in y_{it-j}.

Compared to weighted relative likelihoods, an advantage of the logistic weights is that it allows to specify switching variables in a way that leads to the regimes the econometrician is interested in in a specific application. For instance, if one is interested in how the effects of the shocks vary along with business cycle fluctuations, y_{it-j} may be set as a lagged output gap variable. STVAR models with logistics weights are also easier to estimate than those with the transition weights determined by weighted relative likelihoods of the regimes. A disadvantage is that the empirical results depend highly on the choice of the switching variable, and only the level of the switching variable affects the transition weights.

Multinomial logit transition weights

The logistic transition weights can be generalized to multinomial logit weights that accommodate more than two regimes as well as multiple lags of multiple switching variables as regressors in the logit sub model. The generality, however, comes at the cost of significantly more difficult estimation in the practice and loss of the intuitive interpretations of the parameters of the transition function. With $M \geq 2$ regimes, we specify the multinomial logit weights as

$$\alpha_{m,t} = \frac{\exp\{\gamma_m' z_{t-1}\}}{\sum_{n=1}^{M} \exp\{\gamma_n' z_{t-1}\}}, \quad m = 1, ..., M, \tag{12}$$

where z_{t-1} is an $(k \times 1)$ $\mathcal{F}_{t-1}$-measurable vector containing the (lagged) switching variables and a constant term, γ_m, $m = 1, ..., M-1$, are $(k \times 1)$ coefficient vectors, and the last one is normalized as $\gamma_M = 0$ $(k \times 1)$ to facilitate identification.[2] Denote the set of switching variables as $I \subset \{1, ..., d\}$ (with the indices in I corresponding to the ordering of the variables

[2] Burgard, Neuenkirch, and Nöckel (2019) specify the mixing weights of their mixture VAR in a similar fashion, but unlike us, they allow for exogenous switching variables.

in y_t) and assume that $\tilde{p} \in \{1, ..., p\}$ lags are included in the transition weights. We assume

$$z_{t-1} = (1, \tilde{z}_{\min\{I\}}, ..., \tilde{z}_{\max\{I\}}), \quad \tilde{z}_j = (y_{it-1}, ..., y_{it-\tilde{p}}), \quad i \in I. \tag{13}$$

So $k = 1 + |I|\tilde{p}$ where $|I|$ is the cardinality of the set I (i.e., the number of elements in I). For instance, if the switching variables are the first two variables in y_t, $I = \{1, 2\}$ and only the first lag is included, $\tilde{p} = 1$, we have $z_{t-1} = (1, y_{1t-1}, y_{2t-1})$.

The specification implies

$$\log \frac{\alpha_{m,t}}{\alpha_{M,t}} = \gamma_m' z_{t-1}, \quad m = 1, ..., M-1. \tag{14}$$

Our specification assumes that all the lags up to the lag $\tilde{p}$ are included in the transition weights. The inclusion of only some specific lags in the transition weights is, however, accommodated by imposing constraints on the parameters $\alpha \equiv (\gamma_1, ..., \gamma_{M-1})$ $((M-1)k \times 1)$. Specifically, **sstvars** assumes constraints of the form

$$\alpha = R\xi + r, \tag{15}$$

where R is a known $((M-1)k \times l)$ constraint matrix, r is a known $((M-1)k \times 1)$ constant, and ξ is an unknown $(l \times 1)$ parameter. For instance, by assuming that R is a matrix of zeros, the weight parameter α can be constrained to a known constant.

The logistic weights discussed in the previous section, with y_{it-j} as the switching variable for some lag $j \in \{1, ..., p\}$ and $i \in \{1, ..., d\}$, are obtained as a special case as follows. Assume $M = 2$, $\tilde{p} = j$, and $I = \{i\}$, so that $z_{t-1} = (1, y_{it-1}, ..., y_{it-j})$. Then, impose the constraints $r = 0$ and

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix} \quad (j+1 \times 2), \tag{16}$$

so $\xi = (\gamma_{1,1}, \gamma_{j+1,1})$, where $\gamma_{l,m}$ is the lth element of γ_m.

A direct calculation shows that the "scale parameter" is $-\gamma_{j+1,1)}$ and the "location parameter" is $\frac{\gamma_{1,1}}{-\gamma_{j+1,1}}$. The linear constraints (15) do not, however, enable to constrain the location parameter $\frac{\gamma_{1,1}}{-\gamma_{j+1,1)}}$ to a specific value while leaving the scale parameter $-\gamma_{(j+1),1)}$ unconstrained (or vice versa), or to constrain the scale parameter strictly positive. Therefore, it is more convenient to use the logistic weights and parametrization discussed in Section 2.3.2 when only two regimes and one lag of one switching variable are used.

Exponential transition weights

Exponential transition weights (see, e.g., Teräsvirta 1994) vary according to the level of the switching variable, which we assume to be a lagged endogenous variable. Similarly to the logistic transition weights discussed in Section 2.3.2, the exponential weights depend on a location parameter c and a scale parameter γ that determine the mid point of the transition curve and smoothness of the transitions, respectively. But instead of logistic transition function, we consider an exponential transition function.

Specifically, we assume $M = 2$ and define the exponential transition weights as

$$\alpha_{1,t} = 1 - \alpha_{2,t}, \tag{17}$$
$$\alpha_{2,t} = 1 - \exp\{-\gamma(y_{it-j} - c)^2\} \tag{18}$$

where y_{it-j} is the jth lagged observation ($j \in \{1, ..., p\}$) of the ith variable ($i \in \{1, ..., d\}$), $c \in \mathbb{R}$ is a location parameter, and $\gamma > 0$ is a scale parameter. The location parameter c determines the value of the (lagged) switching variable when the process is completely in first regime, i.e., $\alpha_{1,t} = 1$ and $\alpha_{2,t} = 0$. The closer y_{it-j} is to c, the greater the weight of the first regime is. Conversely, when the deviation of y_{it-j} from c increases, the weight of the second regime increases (and the weight of the first regime decreases). The scale parameter γ, in turn, determines the smoothness of the transitions (smaller γ implies smoother transitions), and it is assumed strictly positive so that $\alpha_{2,t} \in [0, 1]$ for all y_{it-j}.

Threshold transition weights

Threshold transition weights assume discrete regime switches such that the regime switches when the level of the switching variable exceeds or falls below a threshold value. This type model nonlinear VARs are often referred to as Threshold VAR (TVAR) models (Tsay 1998) or self-exciting TVAR models (due to the endogenous switching-variable). We interpret the TVAR model as a special case of the STVAR models, despite of the regime switches being discrete rather than smooth.

For a model with $M > 1$ regimes, consider the $M - 1$ threshold values $r_1, ., .., r_{M-1} \in \mathbb{R}$ such that $r_1 < \cdots < r_{M-1}$, and suppose the switching variable is $i \in \{1, ..., d\}$ with the lag $j \in \{1, ..., p\}$. The transition function is defined as

$$\alpha_{m,t} = \begin{cases} 1 & \text{if } \ r_{m-1} < y_{it-j} \leq r_m, \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

where $r_0 \equiv -\infty$, $r_M \equiv \infty$, and $m = 1, ..., M$. In other words, at each t, the model defined in Equations (1)-(3) and (19) reduces to a linear VAR corresponding to one of the regimes that is determined according to the level of the switching variable y_{it-j}.

Compared to smoothly varying transition weights, threshold transition weights have the advantage that the resulting model is easier to estimate in practice and the regimes have clearer interpretations. An obvious disadvantage is the inability to capture gradual shifts between the regimes or more complex regime-switching dynamics that depend on other factors than just on the level of the switching variable.

3. Structural STVAR models

The STVAR models specified in (1)-(3) directly incorporates the orthogonal, identically and independently distributed structural shocks e_t. The shocks are identified by finding an impact matrix B_t that satisfies Equation (3) for all t and recovers the shocks of interest. There are in general multiple solutions to B_t, since there are d^2 variables but only $d(d + 1)/2$ unique equations in (3). Therefore, further restrictions are required for identification of the shocks. Currently, **sstvars** supports two types of identification methods: recursive identification by the

lower-triangular Cholesky decomposition and identification by conditional heteroskedasticity. Other identification methods will be likely added the package in the future.

3.1. Recursive identification

A conventional way of identifying the shocks is to impose restrictions on the impact responses of variables. A commonly applied identification is to assume a recursive lower-triangular structure on B_t, implying that B_t is obtained as the Cholesky decomposition of the conditional covariance matrix $\Omega_{y,t}$. The recursive identification is straightforward to apply and it allows many of the impact responses to vary in time but constraints many of them to zero. This is particularly disadvantageous if the shock of interest is ordered last or almost last (which is typically the case in small-scale monetary policy shock applications), as then the impact responses of many of the variables to the shock of interest are zero, and therefore, time-invariant.

With our specification of the STVAR model, recursive identification does not, however, allow to impose over-identifying restrictions on the impact matrix B_t. This is because there does not exist a direct parametrization of a lower-triangular B_t such that the conditional covariance matrix $\Omega_{y,t}$ is a weighted sum of the regime-specific covariance matrices with time-varying weights.

3.2. Identification via heteroskedasticity

An alternative identification method proposed by Lütkepohl and Netšunajev (2017) for structural VARs with smooth transitions in variances (see also the seminal paper by Rigobon 2003) identifies the shocks by simultaneously diagonalizing the covariance matrices $\Omega_1, ..., \Omega_M$. This restricts the relative impact responses of the variables to be constant over time (for each shock) but does not necessarily require any zero restrictions. Since Lütkepohl and Netšunajev (2017) assume only two regimes and do not normalize the conditional covariance matrix of the structural error to a constant, their specification does not directly apply to our model. Therefore, we adopt the more suitable specification of Virolainen (2021), and decompose the covariance matrices as

$$\Omega_m = W\Lambda_m W', \quad m = 1, ..., M, \tag{20}$$

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 (20) always exists (Muirhead 1982, Theorem A9.9), but for $M > 2$ its existence requires that the matrices $\Omega_m\Omega_1^{-1}$ share the common eigenvectors in W. This is, however, testable.

The impact matrix is then obtained as

$$B_t = W\left(\sum_{m=1}^{M} \alpha_{m,t}\Lambda_m\right)^{1/2}, \tag{21}$$

where $\Lambda_1 = I_d$. The shocks are identified up to ordering and sign if none of the pairs of λ_{mi}, $i = 1, ..., d$, is identical for all $m = 2, ..., M$. Assuming that his condition is satisfied, the shocks can be labelled according to the unrestricted impact responses on B_t, and if necessary, further economically motivated restrictions can be imposed. The additional economic restrictions on

the impact matrix are testable, as they are overidentifying. See Virolainen (2021) for a more detailed discussion on the identification and labeling of the shocks.

Shocks identified by heteroskedasticity impose constant relative impact responses for the variables, making them unsuitable for some applications concerned with time-varying impulse response functions. Similarly to the recursive identification, this method is straightforward to apply, but unlike the recursive identification, it does not necessarily require any zero constraints on the impact responses. Compared to the recursive identification, identification by heteroskedasticity is therefore particularly advantageous when the recursive identification would imply that the shock of interest is order last. In this case, the assumption of time-invariant relative impact responses is less restrictive than the zero restrictions of the recursive identification. In contrast, the recursive identification is particularly appealing when the shock of interest is ordered first (or almost first), as then the impact responses to the shock of interest can vary freely in time.

Bacchiocchi and Fanelli (2015), Bacchiocchi, Castelnuovo, and Fanelli (2016), and Angelini, Bacchiocchi, Caggiano, and Fanelli (2019) propose alternative way of identifying the shocks by combining heteroskedasticity with zero restrictions on the impact matrices. Their method allows some of the impact responses vary in time also relative to the other variables, but the required untestable zero constraints may challenging to justify economically. Nevertheless, since their identification method is not directly applicable to the STVAR models specified in Section 2, we do not discuss it further.

4. Estimation

The parameters of the reduced form STVAR model are collected to the vector

$$\boldsymbol{\theta} = (\phi_{1,0}, ..., \phi_{m,0}, \varphi_1, ..., \varphi_M, \sigma, \alpha, \nu), \tag{22}$$

where $\varphi_m = (\text{vec}(A_{m,1}), ..., \text{vec}(A_{m,p}))$, $m = 1, ..., M$, $\sigma = (\text{vech}(\Omega_1), ..., \text{vech}(\Omega_M))$, α contains the transition weight parameters, ν is the degrees of freedom parameter (which is dropped for Gaussian models), vec is a vectorization operator that stacks the columns of a matrix on top of each other, and vech stacks the columns of a matrix from the main diagonal downwards (including the main diagonal). Structural models identified by heteroskedasticity assume $\sigma = (\text{vec}(W), \lambda_m, ..., \lambda_M)$, $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$, $m = 2, ..., M$, whereas recursively structural models use the same parameter vector as reduced form models.

If relative stationary densities are used as transition weights, $\alpha = (\alpha_1, ..., \alpha_{M-1})$ (α_M is not included because it is obtained from the constraint $\sum_{m=1}^{M} \alpha_m = 1$), where we assume, for identification, that $\alpha_1, ..., \alpha_{M-1}$ are in a decreasing order. For exponential and logistic transition weights, $\alpha = (c, \gamma)$, for multinomial logit transition weights $\alpha = (\gamma_1, ..., \gamma_{M-1})$ (γ_M is not included because $\gamma_M = 0$ is assumed for identification), and with trhesold transition weights, $\alpha = (r_1, ..., r_{M-1})$.

4.1. Log-likelihood function

sstvars employs the method of maximum likelihood (ML) for estimating the parameters of the STVAR models. Indexing the observed data as $y_{-p+1}, ..., y_0, y_1, ..., y_T$, the conditional

log-likelihood function conditional on the initial values $\boldsymbol{y}_0 = (y_{-p+1}, ..., y_0)$ is given as

$$L_t(\boldsymbol{\theta}) = \sum_{t=1}^{T} l_t(\boldsymbol{\theta}) = \sum_{t=1}^{T} \log d_d(y_t; \mu_{y,t}, \Omega_{y,t}, \nu). \tag{23}$$

where $d_d(y_t; \mu_{y,t}, \Omega_{y,t}, \nu)$ is the d-dimensional conditional density of the process, conditional on $\mathcal{F}_{t-1}$, at time t, given in Section 2.2. When the conditional distribution is Gaussian, the degrees of freedom parameter ν is dropped from the right side of (23). The ML estimator of $\boldsymbol{\theta}$ maximizes the log-likelihood function $L_t(\boldsymbol{\theta})$ over the parameter space specified in the below assumption.

We summarize the constraints imposed on the parameter space in the following assumption.

Assumption 1 *The true parameter value $\boldsymbol{\theta}_0$ is an interior point of $\boldsymbol{\Theta}$, which is a compact subset of $\{\boldsymbol{\theta} = (\phi_{1,0}, ..., \phi_{m,0}, \varphi_1, ..., \varphi_M, \sigma, \alpha, \nu) \in \mathbb{R}^{M(d+d^2p+d(d+1)/2)} \times S \times (2, \infty) : \Omega_m$ is positive definite for all $m = 1, ..., M$, and Condition 1 holds. $\}$.*

Above, $S = (0, 1)^{M-1}$ for relative densities transition weights, $S = \mathbb{R} \times (0, \infty)$ for logistic and exponential weights, $S = \mathbb{R}^{(M-1)k}$ for multinomial logistic weights, and $S = \mathbb{R}^{M-1}$ for threshold weights. As noted before, Condition 1 is not necessary, but it ensures stationarity and ergodicity of the process. In estimation, we impose the more easily verified Condition 2, however, and the sufficient condition can be check after the estimation, if necessary. As noted in Section 2.1.1, the sufficient condition is in practice likely satisfied if the necessary condition is satisfied and not very close to being violated. Given that under Condition 1 the process is ergodic stationary, there is no particular reason to believe that the standard asymptotic results of consistency and limiting Gaussian distribution would not apply to the ML estimator.

4.2. Two-phase estimation procedure

Finding the ML estimate amounts maximizing the log-likelihood function (23) over a high dimensional parameter space satisfying the constraints summarized in Assumption 1. Due to the complexity of the log-likelihood function, numerical optimization methods are required. The maximization problem can be challenging in practice due to the dependence of the transition weights 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.

Therefore, we follow Meitz *et al.* (2023) and Virolainen (2022b) and employ a two-phase estimation procedure that is run for a large number of times. In the first phase, a genetic algorithm is used to find parameter values (hopefully) near local maximums. Since genetic algorithms tend to converge slowly near local solutions, a gradient based variable algorithm (Nash 1990, algorithm 21, implemented by R Core Team 2022) is ran for each of the starting values, resulting in a number of alternative local solutions. Some of the estimation rounds may end up in saddle points or near-the-boundary points that are not local solutions, and some of the local solutions may be inappropriate for econometric inference (for instance, there might be only a few observations from some of the regimes). After the estimation rounds have been ran, the researcher can choose the local solution that maximizes the log-likelihood among the appropriate local solutions. Inappropriate solutions are automatically filtered by **sstvars**,

but this functionality can also be turned off and the researchers can use estimates based any estimation round. The R package **sstvars** employs a modified genetic algorithm that works similarly to the one described in the R packages **uGMAR** (Virolainen 2018b) and **gmvarkit** (Virolainen 2018a) (the genetic algorithm and implemented in former is briefly described in Virolainen 2022b). See Virolainen (2022a, Chapter 3) for a related discussion on complex numerical estimation problems using the two-phase procedure.

4.3. Examples of unconstrained estimation

In this section, we demonstrate how to estimate STVAR models with **sstvars**. In the examples, we only consider $p = 1$ models for simplicity and merely because then the code outputs fit in the margins better. This order may not be the best in the modeling perspective, however.

In **sstvars**, the STVAR models are defined as class `stvar` S3 objects, which can be created with given parameter values using the constructor function `STVAR` (see Section 7) or by using the estimation function `fitSTVAR`, which estimates the parameters and then builds the (reduced form) model. Structural models are estimated based on a reduced form model with the function `fitSSTVAR`. For estimation, `fitSTVAR` needs to be supplied with a multivariate time series and the arguments specifying the model. The necessary arguments for specifying the model include the autoregressive order `p`, the number of regimes `M`, the transition weight function `weight_function`, with some weight functions the switching variable(s) `weightfun_pars`, and the conditional distribution `cond_dist`.

Additional arguments may be supplied to `fitSTVAR` in order to specify, most importantly, how many estimation rounds should be performed (`nrounds`) 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 regimes is large or there are many variables in the data, 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. Another important thing to know about estimation is that the estimation algorithm performs very poorly if some of the AR coefficients are very large, substantially larger than one. This means that you need to scale each component time series so that they vary approximately in the same magnitude. For instance, typically in macroeconomic time series, log-differences should be multiplied by hundred. If the suitable scales are not obvious, you can try out different scales and estimate linear VARs with your favorite package to see whether the AR coeffients are in a reasonable range. When a suitable scale is found, proceed to the STVAR models.

We illustrate the use of **sstvars** 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 STVAR($p = 1, M = 2$)

model with Student's t conditional distribution and logistic transition weights by performing 24 estimation rounds with 8 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 and the more there are regimes, the more estimation rounds is required. The model in our example is easy to estimate, as it is small in dimension and order.** We set the switching variable to be the first lag of the second variable, i.e., the GDP deflator by setting argument `weightfun_pars=c(2, 1)`.

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> library(sstvars)
R> data("gdpdef", package="sstvars")
R> fit12t <- fitSTVAR(gdpdef, p=1, M=2, weight_function="logistic",
+    weightfun_pars=c(2, 1), cond_dist="Student", nrounds=24, ncores=8,
+    seeds=1:24)

Using 8 cores for 24 estimations rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=31s
Results from the genetic algorithm:
The lowest loglik:  -283.893
The largest loglik: -253.864
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=03s
Results from the variable metric algorithm:
The lowest loglik:  -276.669
The largest loglik: -250.236
Filtering inappropriate estimates...
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).

Because the log-likelihood function is highly multimodal, and the estimation algorithm is ran a large number of times, it produces a set of local solutions, possibly representing various modes in the log-likelihood function. Some of the local solutions may be inappropriate for econometric inference, for instance, because they contain a near-singular error term covariance matrix or regimes that have only a very small number of observations generated (even partially) from them. Such inappropriate solutions, i.e., estimates that are not solutions of interest, are filtered automatically by **sstvars**. Specifically, solutions that incorporate a near-singular error term covariance matrix (any eigenvalue less than 0.002), any modulus "bold A" eigenvalues larger than 0.9985 (indicating the necessary condition for stationarity is close to break), or transition weights such that they are close to zero for almost all t for at least one

regime. With relative densities transition weights, also solutions in which a weight parameter estimate is close to zero are filtered out.

Automatic filtering can be turned off with the argument `filter_estimates=FALSE`. Then, the various local solutions can be easily examined by hand by using the function `alt_stvar` and adjusting its argument `which_largest` or `which_round`. Note that `alt_stvar` can be used also when the automatic filtering is turned on, since the estimates from all of the estimation rounds are stored in the class `'stvar'` object returned by `fitSTVAR`.

The estimates can be examined with the `print` method.

```
R> print(fit12)

logistic Student STVAR model, reduced form model no AR_constraints, no mean_constraints,
  p = 1, M = 2, d = 2, #parameters = 21, #observations = 243 x 2
  Switching variable: GDPDEF with lag 1.

Regime 1
Degrees of freedom: 7.70 (for all regimes)
Regime means: 0.71, 0.49

   Y     phi0          A1                  Omega        1/2
1 y1 = [ 0.63 ] + [  0.35 -0.35 ] y1.1 + [  0.37 0.00 ]     eps1
2 y2   [ 0.14 ]   [  0.06  0.62 ] y2.1   [  0.00 0.03 ]     eps2

Regime 2
Weight params: 1.22 (location), 5.01 (scale)
Regime means: 0.77, 1.76

   Y     phi0          A1                  Omega         1/2
1 y1 = [ 2.41 ] + [  0.13 -0.99 ] y1.1 + [  1.29 -0.06 ]     eps1
2 y2   [ 0.67 ]   [ -0.04  0.64 ] y2.1   [ -0.06  0.19 ]     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

$$y_t = \varphi_{m,0} + A_{m,1}y_{t-1} + ... + A_{m,p}y_{t-p} + \Omega_m^{1/2}e_t. \tag{24}$$

The other statistics are listed above the formula: the degrees of freedom parameter estimate, the unconditional means of the regimes, and estimates of the transition weight parameters. Based on the estimate of the location parameter and the conditional means, the second regime accommodates periods of high inflation and the first regime thereby periods of lower inflation.

A more detailed printout is obtained with the `summary` method as follows:

```
R> summary(fit12)

logistic Student STVAR model, reduced form model no AR_constraints, no mean_constraints,
  p = 1, M = 2, d = 2, #parameters = 21, #observations = 243 x 2
  Switching variable: GDPDEF with lag 1.
```

```
loglik/T: -1.03, AIC: 2.23, HQIC: 2.35, BIC: 2.53

Regime 1
Degrees of freedom: 7.70 (for all regimes)
Moduli of 'bold A' eigenvalues:  0.49, 0.49
Cov. matrix 'Omega' eigenvalues: 0.37, 0.03
Regime means: 0.71, 0.49
Regime sdevs: 0.66, 0.24

   Y     phi0          A1                   Omega        1/2
1 y1 = [ 0.63 ] + [  0.35 -0.35 ] y1.1 + [  0.37 0.00 ]     eps1
2 y2   [ 0.14 ]   [  0.06  0.62 ] y2.1   [  0.00 0.03 ]     eps2

Error term correlation matrix:
      [,1]  [,2]
[1,] 1.000 0.028
[2,] 0.028 1.000

Regime 2
Moduli of 'bold A' eigenvalues:  0.71, 0.06
Cov. matrix 'Omega' eigenvalues: 1.29, 0.18
Weight params: 1.22 (location), 5.01 (scale)
Regime means: 0.77, 1.76
Regime sdevs: 1.32, 0.59

   Y     phi0          A1                   Omega         1/2
1 y1 = [ 2.41 ] + [  0.13 -0.99 ] y1.1 + [  1.29 -0.06 ]     eps1
2 y2   [ 0.67 ]   [ -0.04  0.64 ] y2.1   [ -0.06  0.19 ]     eps2

Error term correlation matrix:
      [,1]  [,2]
[1,]  1.00 -0.12
[2,] -0.12  1.00

Print approximate standard errors with the function 'print_std_errors'.
```

The above summary printout shows additional information compared to the print method, including moduli of the eigenvalues of the companion form AR matrices (to assess how close Condition 2 is to break), eigenvalues of the error term covariance matrices (to assess how close they are to being non-positive definite), marginal standard deviations of the variables in each regime, error term correlation matrices, and the log-likelihood as well as the values of the information criteria divided by the number of observations $T - p$.

Approximate standard errors can be printed with the function `print_std_errors`, which prints the standard errors in the same form as the print method prints the estimates. There is no standard error for the intercepts if mean parametrization is used (by setting `parametrization = "mean"` in `fitSTVAR`) and vice versa. In order to obtain standard errors for the regime-

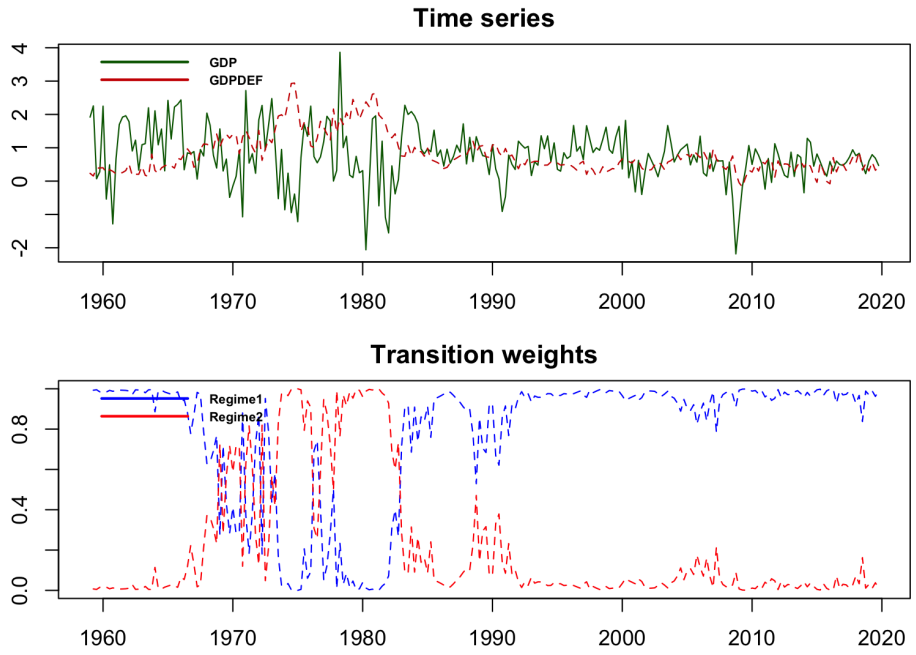

Figure 1: The figure produced by the command `plot(fit12)`. On the top, a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 to 2019Q4. On the bottom, the estimated transition weights of the STVAR model `fit12gs` fitted the series.

wise unconditional means or intercepts, one can easily swap between the mean and intercept parametrizations with the function `swap_parametrization`. Note that approximate standard errors are based on the mere assumption of the standard Gaussian asymptotic distribution of the estimator. Hence, they should be interpreted with caution.

4.4. Further examination of the estimates

In addition to examining the summary printout, it is often useful to visualize the model by plotting the transition weights together with the time series. That is exactly what the plot method for GSMVAR models does. The following command creates the time series plot along with estimated mixing weights:

```
R> plot(fit12, type="series")
```

The resulting plot is presented in Figure 1. The figure confirms our earlier observation: the second regime is dominant during the periods of high inflation, particularly in the volatile periods of 1970's and 1980's.

It is also sometimes interesting to examine the time series of (one-step) conditional means of the process along with the time series the model was fitted to. This can be done conveniently with the function by setting the argument `plot_type="cond_mean"` in the plot method. This plot depicts the contribution of each regime to the conditional mean of the process and how close the conditional mean is to the observed series in each point of time.

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

sstvars 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

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} \approx \frac{f(\boldsymbol{\theta} + \boldsymbol{h}^{(i)}) - f(\boldsymbol{\theta} - \boldsymbol{h}^{(i)})}{2h}, \quad h > 0, \tag{25}$$

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 example, the following code calculates the first order condition for the G-StMVAR model `fit12`:

```
R> get_foc(fit12)

 [1]  4.262025e-04  4.048578e-04  1.281037e-04 -5.780028e-04
 [5]  3.221411e-04  1.776783e-04  2.645895e-04 -3.844178e-04
 [9] -2.174971e-05  1.380419e-04  1.664494e-04 -8.551879e-04
[13]  1.246268e-04 -3.826628e-04 -2.131233e-03 -2.536638e-06
[17]  1.432667e-04  1.818326e-04  3.602239e-04  1.000681e-05
[21] -3.700270e-05
```

and the following code calculates the second order condition:

```
R> get_soc(fit12)

 [1] -1.494320e-01 -2.682544e-01 -1.047386e+00 -6.177532e+00
 [5] -9.934297e+00 -1.654975e+01 -5.366908e+01 -7.191557e+01
 [9] -1.215715e+02 -1.673756e+02 -1.859815e+02 -2.718937e+02
[13] -3.775367e+02 -3.834231e+02 -6.567316e+02 -9.911009e+02
[17] -1.227186e+03 -1.377650e+03 -8.775102e+03 -9.731932e+03
[21] -3.895717e+04
```

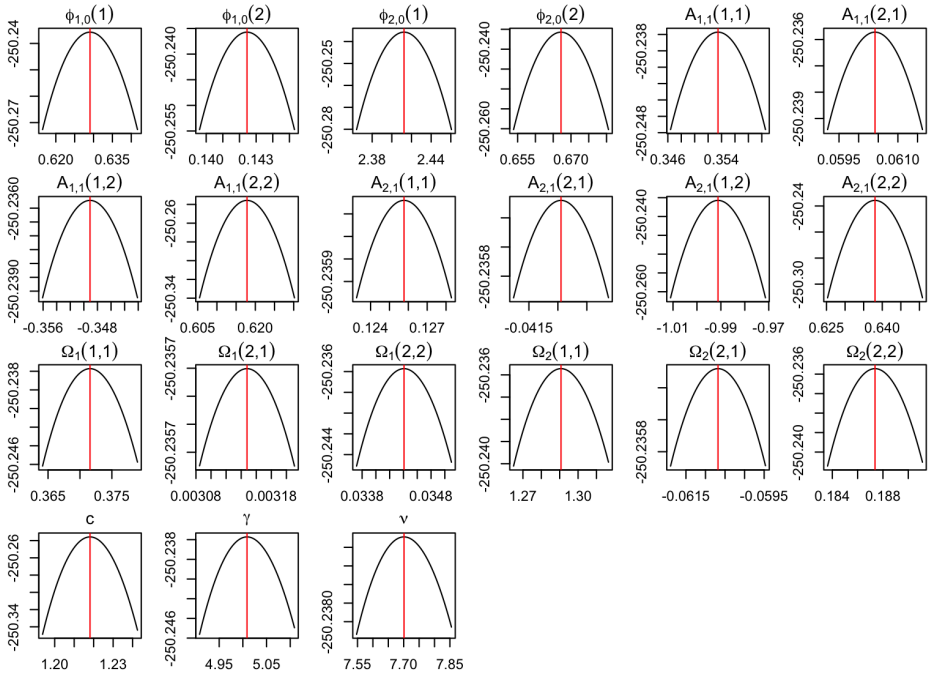

Figure 2: The figure produced by the command `profile_logliks(fit12, scale=0.02, precision=200)`. The graphs of the profile log-likelihood functions of the logistic Student's t STVAR model drawn about the estimate. The red vertical lines denote the estimate.

All eigenvalues of the Hessian matrix are negative, which points to a local maximum, and the gradient of the log-likelihood function is close to zero. The gradient is not exactly zero, 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 **sstvars**, 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-StMVAR model `fit12`:

```
R> profile_logliks(fit12, scale=0.02, precision=200)
```

The resulting plot is presented in Figure 2.

The output shows that the estimate's accuracy is reasonable, as changing any individual parameter value marginally would not increase the log-likelihood much. 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. The argument `which_pars` is used to specify the parameters whose profile log-likelihood functions should be plotted. This argument is particularly useful when

creating as many plots as there are parameters in the model to a single figure would cause the individual plots to be very small. In such a case, profile log-likelihood functions for subsets of the parameters can be plotted separately by specifying this argument accordingly.

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 **sstvars**, the best way to increase the reliability that the found estimate is the global maximum (among the appropriate solutions), is to run more estimation rounds by adjusting the argument `nrounds` of the estimation function `fitSTVAR`.

If the model is very large, a very large number of estimation rounds may be required to find the global maximum. If there are two regimes in the model, p is reasonable, and the dimension of the time series at most four, the required number of estimation rounds typically varies from several hundred to several thousand depending on the model and the data. In the simpler models, less estimation rounds are required. In the larger models, and in particular if $M > 2$ or $d > 4$, a significantly large number of estimation rounds may be required obtain the MLE. Another thing that makes the estimation more challenging, are exotic parameter constraints that do not reduce the dimension of the parameter much. Constraints that greatly reduce complexity of the parameter space (such as constraining the autoregressive matrices to be identical in all regimes[3]), on the other hand, make the estimation easier, and reliable estimation of such models thereby require less estimation rounds. Constrained estimation is discussed in Section 4.6.

4.5. Estimation of structural STVAR models

As explained, **sstvars** currently supports two types of structural models: structural models identified recursively by the lower triangular Cholesky decomposition and structural models identified by conditional heteroskedasticity. In either case, the structural models are estimated with the function `fitSSTVAR` based on preliminary estimates from a reduced form model. If the structural model is not overidentifying, which is always the case for recursively identified models as well as for models identified by heteroskedasticity when there are two regimes and further constraints are not imposed on the impact matrix, there is no need for estimation but at most for a reparametrization of the model. In any case, the `fitSSTVAR` constructs the structural model appropriately.

To exemplify, we first create a recursively identified structural model based on the reduced form model `fit12` by setting the argument `identification="recursive"`. Then, we will study an example of a structural model identified by heteroskedasticity. The following code builds the recursively identified model:

```
R> fit12rec <- fitSSTVAR(fit12, identification="recursive")
```

Since the parametrization did not change nor was any estimation required, `fit12rec` is essentially the reduced form model `fit12` with has the property that can be used as a structural model in structural analysis such as for estimating the generalized impulse response functions.

The following code creates a structural model identified by heteroskedasticity based on the reduced form model `fit12` and then prints it:

[3]Models constrained in this way can often be reliably estimated with a reasonable number of estimation rounds even when $M > 2$

```
R> fit12het <- fitSSTVAR(fit12, identification="heteroskedasticity")
R> print(fit12het)

logistic Student STVAR model, identified by heteroskedasticity, no AR_constraints,
no mean_constraints, no B_constraints,
  p = 1, M = 2, d = 2, #parameters = 21, #observations = 243 x 2
  Switching variable: GDPDEF with lag 1.

Regime 1
Degrees of freedom: 7.70 (for all regimes)
Regime means: 0.71, 0.49

   Y     phi0          A1                   Omega        1/2
1 y1 = [ 0.63 ] + [  0.35 -0.35 ] y1.1 + [  0.37 0.00 ]     eps1
2 y2   [ 0.14 ]   [  0.06  0.62 ] y2.1   [  0.00 0.03 ]     eps2

Regime 2
Weight params: 1.22 (location), 5.01 (scale)
Regime means: 0.77, 1.76

   Y     phi0          A1                   Omega         1/2
1 y1 = [ 2.41 ] + [  0.13 -0.99 ] y1.1 + [  1.29 -0.06 ]     eps1
2 y2   [ 0.67 ]   [ -0.04  0.64 ] y2.1   [ -0.06  0.19 ]     eps2

Structural parameters:
        W             lamb2
1 [  0.17  0.59 ]   [  5.67 ]
2 [ -0.18  0.06 ] , [  3.29 ]

The impact matrix is subject to 0 zero constraints and 0 sign constraints.
```

Since the structural model identified by heteroskedasticity is not overidentified, no estimation was performed but merely a repameterization. The estimates for the structural parameters W and $\lambda_2, ..., \lambda_M$ are presented at the bottom of the printout.

If the structural model is identified by heteroskedasticity, additional restrictions can be imposed on the impact matrix by setting them in the argument `B_constraints`. A structural model identified by heteroskedasticity is overidentifying also when there are more than two regimes in the model, as then the matrix decomposition employed in the identification does not always exist. In either case, the structural model needs be estimated, which is performed in `sstvars` based on preliminary estimates obtained either from a reduced form model or from a structural model. The estimation is performed with the function `fitSSTVAR`, which implements a two-phase estimation procedure in which a robust estimation method is used in the first phase and a variable algorithm in the second phase. The default option for the robust method is Nelder-Mead algorithm implemented by R Core Team (2022) in the function `optim` of the package `stats`.

It is important to note that if the initial estimates are far from the ML estimate of the overidentified model, the resulting solution is likely local only due to the high multimodality

of the log-likelihood function. However, it is not often very appealing to impose overidentified constraints that far from the unrestricted estimates in the first place. But in any case, since the estimation may be unrealiable if the restricted ML estimate is far from the unrestricted ML estimate, we recommend using our package to estimate only such overidentifying structural models in which the unrestricted estimate is close to satisfying the imposed constraints.

The exemplify, we estimate a structural model identified by heteroskedasticity based on the structural model `fit12het` by setting the argument `identification="heteroskedasticity"` and imposing the constraint that the second element of the second column of the impact matrix is zero (the above estimates of W show that the corresponding unrestricted estimate is close to zero). The zero constraint is imposed by setting the argument `B_constraints` as matrix such that the second element of the second column is zero and all other elements are `NA`. Sign constraints can be set similarly by setting the corresponding elements to `1` or `-1` (or any other strictly positive of negative value). The following code estimates the model:

```
R> fit12hetb <- fitSSTVAR(fit12, identification="heteroskedasticity",
+    B_constraints=matrix(c(NA, NA, NA, 0), nrow=2))
R> print(fit12het)

The log-likelihood of the supplied model:    -250.236
Constrained log-likelihood prior estimation: -260.164
The log-likelihood after robust estimation:  -250.556
The log-likelihood after final estimation:   -250.486
```

The log-likelihoods of the original model, the initial estimates of the constrained model, the estimates after the robust estimation, and the final estimates are printed. If the log-likelihood of after final estimation is bad compared to the log-likelihood of the original model, the estimation is likely unreliable. The command `print(fit12hetb)` prints the estimated overidentified model (we ommit the printout for brevity).

After creating a structural model identified by heteroskedasticity, the columns of W can be reordered with the function `reorder_W_columns` which also reorders all λ_{mi} accordingly (and hence the resulting model will coincide with the original reduced form model). Also, all signs of any column of W can be swapped with the function `swap_W_signs`.

4.6. Constrained estimation

sstvars supports constrained ML estimation of the STVAR models, including several types of constraints. Linear constraints can be imposed on the autoregressive matrices (argument `AR_constraints`), unconditional means of the regimes can be constrained equal across (groups of) regimes (argument `mean_constraints`), and weight function parameters can be constrained to a fixed value or linear constraints can be impose on them (argument `weight_constraints`). Following sections give examples of constrained estimation imposing some of these constraints.

Linear constraints on the autoregressive parameters

Imposing linear constraints on the autoregressive parameters of a STVAR model is straightforward in **sstvars**. The constraints are expressed in a somewhat general form which allows to

impose a wide class of constraints but one needs to take the time to construct the constraint matrix carefully for each particular case.

We consider constraints of form

$$(\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M) = \boldsymbol{C}\boldsymbol{\psi}, \tag{26}$$

$$\boldsymbol{\phi}_m = (vec(A_{m,1}), ..., vec(A_{m,p})) \ \ (pd^2x1), \ \ m = 1, ..., M, \tag{27}$$

where $\boldsymbol{C}$ is known (Mpd^2xq) constraint matrix (of full column rank) and $\boldsymbol{\psi}$ is unknown ($qx1$) parameter vector.

To give couple examples, consider the following two common uses of linear constraints: restricting the autoregressive matrices to be the equal across all regimes and constraining some of the AR parameters to zero.

Restricting AR matrices to be the equal across the regimes

To restrict the AR matrices to be equal across the regimes, we want $\boldsymbol{\phi}_m$ to be the same for all $m = 1, ..., M$. The parameter vector $\boldsymbol{\psi}$ ($qx1$) then corresponds to any $\boldsymbol{\phi}_m = \boldsymbol{\phi}$, and therefore $q = pd^2$. For the constraint matrix we choose

$$\boldsymbol{C} = [I_{pd^2} : \cdots : I_{pd^2}]' \ \ (Mpd^2xpd^2), \tag{28}$$

that is, M pieces of (pd^2xpd^2) diagonal matrices stacked on top of each other, because then

$$\boldsymbol{C}\boldsymbol{\psi} = (\boldsymbol{\psi}, ..., \boldsymbol{\psi}) = (\boldsymbol{\phi}, ..., \boldsymbol{\phi}). \tag{29}$$

For instance, if there are two regimes in the model, the appropriate constraint matrix then created as

```
R> p <- 1 # The autoregressive order of the model
R> d <- 2 # Whatever the dimension of the time series is
R> I_pd2 <- diag(p*d^2) # The appropriate diagonal matrix
R> (C1 <- rbind(I_pd2, I_pd2)) # Stack them on top of each other
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
[5,]    1    0    0    0
[6,]    0    1    0    0
[7,]    0    0    1    0
[8,]    0    0    0    1
```

The command

```
R> fit12c1 <- fitSTVAR(gdpdef, p=1, M=2, weight_function="logistic",
+    weightfun_pars=c(2, 1), cond_dist="Student", AR_constraints=C1)
```

would then estimate a logistic Student's t STVAR$(1, 2)$ model with first lag of the second variable as the switching variable such that the AR matrices constrained to be the equal in both regimes. We omit the output for brevity. In practice, you might want to adjust the number of CPU cores used (`ncores`), the of estimation rounds (`nrounds`), and set seeds (`seeds`).

Restricting AR parameters to be the same for all regimes and constraining non-diagonal elements of coefficient matrices to be zero

The previous example shows how to restrict the AR parameters to be the same for all regimes, but say we also want to constrain the non-diagonal elements of coefficient matrices $A_{m,i}$ $(m = 1, ..., M, i = 1, ..., p)$ to be zero. We have the constrained parameter $\boldsymbol{\psi}$ $(qx1)$ representing the unconstrained parameters $(\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M)$, where the restrictions imply $\boldsymbol{\phi}_m = \boldsymbol{\phi} = (vec(A_1), ..., vec(A_p))$ (pd^2x1) and the elements of $vec(A_i)$ $(i = 1, ..., p)$ corresponding to the diagonal are zero.

For illustrative purposes, let's consider a STVAR model with autoregressive degree $p = 2$, number of regimes $M = 2$, and number of time series in the system $d = 2$. Then we have

$$\boldsymbol{\phi} = (A_1(1,1), 0, 0, A_1(2,2), A_2(1,1), 0, 0, A_2(2,2)) \ (8x1) \text{ and} \tag{30}$$

$$\boldsymbol{\psi} = (A_1(1,1), A_1(2,2), A_2(1,1), A_2(2,2)) \ (4x1), \tag{31}$$

where $A_l(i, j)$ is the ijth elements of A_l. By a direct calculation, we can see that choosing the constraint matrix

$$\boldsymbol{C} = \begin{bmatrix} \tilde{\boldsymbol{c}} \\ \tilde{\boldsymbol{c}} \end{bmatrix} \ (Mpd^2x4), \tag{32}$$

where

$$\tilde{\boldsymbol{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ (pd^2x4) \tag{33}$$

satisfies $\boldsymbol{C\psi} = (\boldsymbol{\phi}, ..., \boldsymbol{\phi})$.

The above constraint matrix can be created as

```
R> c_tilde <- matrix(0, nrow=2*2^2, ncol=4)
R> c_tilde[c(1, 12, 21, 32)] <- 1
R> C2 <- rbind(c_tilde, c_tilde)
R> C2
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    1    0    0
[5,]    0    0    1    0
```

```
 [6,]    0    0    0    0
 [7,]    0    0    0    0
 [8,]    0    0    0    1
 [9,]    1    0    0    0
[10,]    0    0    0    0
[11,]    0    0    0    0
[12,]    0    1    0    0
[13,]    0    0    1    0
[14,]    0    0    0    0
[15,]    0    0    0    0
[16,]    0    0    0    1
```

The command

```
R> fit12c2 <- fitSTVAR(gdpdef, p=2, M=2, weight_function="logistic",
+    weightfun_pars=c(2, 1), cond_dist="Student", AR_constraints=C2)
```

would then estimate a logistic Student's t STVAR$(1, 2)$ model with first lag of the second variable as the switching variable such that the AR matrices are constrained to be the equal in both regimes and the off-diagonal elements are restricted to zero. Again, we omit the output for brevity (and you may want to adjust the arguments `nrounds`, `ncores`, and `seeds` when estimating the model in practice).

Constraining the unconditional means of some regimes to be equal

In addition to constraining the autoregressive parameters, **sstvars** allows to constrain the unconditional means of some regimes to be the equal. This feature is, however, only available for models that are parametrized with the unconditional means instead of intercepts (because some of the estimation is always done with mean-parametrization and one cannot generally swap the parametrization when constraints are imposed on means/intercepts). With the mean-parametrization employed (by setting `parametrization="mean"`), one may define groups of regimes that have the same mean parameters using the argument `mean_constraints`. For instance, with three regime model ($M = 3$) the argument `mean_constraints=list(c(1, 3), 2)` sets the unconditional means of the first and third regimes to be the same while allows the second regime to have different mean.

One can also combine linear constraints on the AR parameters with constraining some of the means to be the same. This allows, for instance, to estimate a model in which only the covariance matrix varies in time. To exemplify, the following code estimates a Student's t logistic STVAR$(p = 1, M = 2)$ model such that the unconditional means and autoregression matrices are constrained be the same in both regimes. The resulting model thereby has time-varying covariance matrix but otherwise it is linear.

```
R> fit12c3 <- fitSTVAR(gdpdef, p=1, M=2, weight_function="logistic",
+    weightfun_pars=c(2, 1), cond_dist="Student", AR_constraints=C1,
+    mean_constraints=list(1:2), parametrization="mean")
```

The output is omitted for brevity.

Constraining weight functions parameters

It is also possible to constrain the weight functions parameters α (e.g., the location and scale parameters for logistic models). **sstvars** accommodates two types of alternative constraints on the weight function parameters: linear constraints and fixed values.

Linear constraints
Linear constraints on the weight function parameters are of the form

$$\alpha = R\xi + r, \tag{34}$$

where α $(a \times 1)$ contains the weight funtion parameters, R is a known $(a \times l)$ constraint matrix of full column rank, r is a known $(a \times 1)$ constant, and ξ is an unknown $(l \times 1)$ parameter. The constraint matrix R and the constant r are set in the argument `weight_constraints` as a list of two elements, R in the first element and r in the second element. The number of unknown parameters l is the number of columns of R. For instance, the following argument imposes constraints for the scale and location parameters of a logistic STVAR model (in which $\alpha = (c, \gamma)$) such that the location parameter is the scale parameter divided by two plus 0.3: `weight_constraints=list(R=matrix(c(0.5, 1), nrow=2), r=c(0.3, 0))`.

Fixed values
Imposing the weight function parameters to be known fixed values is very straightforward. In this case, the argument `weight_constraints` should still be a list including the elements R and r, but the former is set to zero, $R = 0$, and the latter is set to the desired fixed values. For instance the following argument imposes constraints for the location and location parameters of a logistic STVAR model (in which $\alpha = (c, \gamma)$) such that the location parameter is 0.3 and the scale parameter is 0.5: `weight_constraints=list(R=0, r=c(0.3, 0.5))`.

4.7. 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. **sstvars**, however, also provides functions for testing the constraints with the likelihood ratio test, Wald test, and Rao's test, whose applicability requires that the ML estimator of the STVAR model has the conventional asymptotic distribution. As noted before, this is a mere assumption, but given the process is ergodic stationary, there is no particular reason to believe that the standard asymptotic results would not hold. 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 Assumption 1). 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

$$LR = 2(\hat{L}_U - \hat{L}_C). \tag{35}$$

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 **sstvars**, the likelihood ratio test can be calculated with the function `LR_test`,

which takes the unconstrained model (a class `'stvar'` object) as its first argument and the constrained model as the second argument.

sstvars implements the Wald test of the null hypothesis

$$A\boldsymbol{\theta}_0 = c, \tag{36}$$

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

$$W = (A\hat{\boldsymbol{\theta}} - c)'[A\mathcal{J}(\hat{\boldsymbol{\theta}})^{-1}A']^{-1}(A\hat{\boldsymbol{\theta}} - c), \tag{37}$$

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 **sstvars**, the Wald test can be calculated with function `Wald_test`, which takes the estimated unconstrained model (as a class `'stvar'` object) as the first argument, the matrix A as the second argument, and the vector c as the third argument.

Rao's test is implemented to the function `Rao_test` (see the function documentation on how to use it).

5. Residual based model diagnostics

sstvars employs residual based diagnostics for assessing the adequacy of the fitted model. Conventional graphical diagnostics can be examined with function `diagnostic_plot`, which plots the residual time series, auto- and crosscorrelation functions of the residuals, auto- and crosscorrelation functions of the squared residuals, and normal quantile-quantile plots as well as histograms of the residuals. The plots can be created for both unstandardized residuals or for standardized residuals by adjusting the argument `standardize` to `FALSE` or `TRUE` according. Using unstandardized residuals is advisable when checking for remaining autocorrelation. But standardized residuals should be used to check for remaining conditional heterokedasticity and to check the model's adequacy to capture the marginal distribution of the series, because the STVAR models are conditionally heteroskedastic and the unstandardized residuals do not take into account the time-varying conditional covariance matrix.

Remaining autocorrelation in the residuals can also be tested with the (adjusted) Portmanteau test, which is implemented to the function `Portmantau_test`. The test can also be applied to standardized squared residuals to test for remaining conditional heteroskedasticity by setting the argument `which_test="het.sked"`. The number of lags that should be taken into account in the test is set with the argument `nlags`.

6. Impulse response analysis

6.1. Generalized impulse response function

The expected effects of the structural shocks in the structural STVAR models generally depend on the initial values as well as on the sign and size of the shock, which makes the

conventional way of calculating impulse responses unsuitable (see, e.g., Kilian and Lütkepohl 2017, Chapter 4). Therefore, we consider the generalized impulse response function (GIRF) (Koop *et al.* 1996) defined as

$$\text{GIRF}(n, \delta_j, \mathcal{F}_{t-1}) = \text{E}[y_{t+n}|\delta_j, \mathcal{F}_{t-1}] - \text{E}[y_{t+n}|\mathcal{F}_{t-1}], \tag{38}$$

where n is the chosen horizon, $\mathcal{F}_{t-1} = \sigma\{y_{t-j}, j > 0\}$ as before, the first term on the right side is the expected realization of the process at time $t+n$ conditionally on a structural shock of sign and size $\delta_j \in \mathbb{R}$ in the jth element of e_t at time t and the previous observations, and the second term on the right side 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 implemented STVAR models, 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 initial values (or 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. Alternatively, one can estimate GIRFs conditional on the initial values being from a specific regime, in which case $\boldsymbol{y}_{t-1}$ should generated from the regime of interest.

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 **sstvars** is presented in FILL IN REFERENCE TO LANNE AND VIROLAINEN.

Because the STVAR models allow to associate specific features or economic interpretations for different regimes, and because shifts in the regime are the source of asymmetries in the impulse responses, it might be interesting to also examine the effects of a structural shock to the transition 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 (38).

In **sstvars**, the GIRF can be estimated with the function `GIRF` which should be supplied with the estimated STVAR model or a STVAR model built with hand-specified parameter values using the function `STVAR`. Structural models can be created based on a reduced form model with the function `fitSSTVAR`. The sign and size of the structural shock can be set with the argument `shock_size`. If not specified, a positive shock with the size of one standard deviation is used; that is, the size is one. Among other arguments, the function may also be supplied with the argument `init_regime` which specifies from which regime the initial values are generated from. 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.

Due to the nonlinear nature of the model, GIRFs estimated from different starting values, or with different sign or magnitude of the shock, generally move the variables differently.

Sometimes it is, however, of interest to scale the impulse responses so that they correspond to instantaneous movement of some specific sign and size of some specific variable. In **sstvars**, this is most conveniently achieved with the arguments `scale`. The argument `scale` can be specified in order to scale the GIRFs to some of the shocks so that they correspond to a specific sign and size of instantaneous response of some specific variable. For a single shock, it should a length three vector where the shock of interest is given in the first element (an integer in $1, ..., d$), the variable according to which the GIRFs should be scaled in the second element (an integer in $1, ..., d$), and the sign and size of the given variable's instantaneous response in the third element (a non-zero real number). If the GIRFs of multiple shocks should be scaled, provide a matrix which has one column for each of the shocks with the columns being the length three vectors described above. Note that if you scale the GIRFs, the scaled GIRFs of transition weights can be outside the interval from zero to one.

Because estimating the GIRF and their confidence intervals 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`.

6.2. Generalized forecast error variance decomposition

Similarly to the conventional impulse response functions are unsuitable for impulse response analysis (due to their inability to capture asymmetries the effects of the shocks), the conventional forecast error variance decomposition is unsuitable for tracking the contribution of each shock to the variance of the forecast errors. We consider the generalized forecast error variance decomposition (GFEVD) (Lanne and Nyberg 2016) that is defined for variable i, shock j, and horizon n as

$$\text{GFEVD}(n, y_{it}, \delta_j, \mathcal{F}_{t-1}) = \frac{\sum_{l=0}^{n} \text{GIRF}(l, \delta_j, \mathcal{F}_{t-1})_i^2}{\sum_{k=1}^{d} \sum_{l=0}^{n} \text{GIRF}(l, \delta_k, \mathcal{F}_{t-1})_i^2}, \tag{39}$$

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 **sstvars**, 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 one of the specific regimes, specified by 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.

`sstvars` also implements a special feature in which for every possible length p history in the data, the GFEVD is estimated for a shock that has the sign and size of the corresponding structural shock recovered from the fitted model. This can be done by setting the argument `use_data_shocks=TRUE`. The GFEVD is then calculated as the average of the GFEVDs obtained from the GIRFs estimated for the data shocks. The plot and print methods can be used as usual for this GFEVD. However, this feature also estimates the contribution of each shock to the variance of the forecast errors at various horizons in specific historical points of time. This can be done by using the plot method with the argument `data_shock_pars`. Note that the arguments `shock_size`, `initval_type`, and `init_regime` are ignored if `use_data_shocks == TRUE`.

6.3. Linear impulse response functions

It is also possible to calculate linear impulse response functions (IRF) based on a specific regime of the estimated model by using the function `linear_IRF`. If the autoregressive dynamics of the model are linear (i.e., either $M = 1$ or mean and AR parameters are constrained identical across the regimes), confidence bounds can be estimated based on a type of a fixed-design wild residual bootstrap method. `sstvars` implements the method proposed Herwartz and Lütkepohl (2014).

7. Building a STVAR model with specific parameter values

The function `STVAR` facilitates building STVAR models without estimation, for instance, in order to simulate observations from a STVAR process with specific parameter values. The function should be supplied at least with the arguments `p`, `M`, and `d` specifying the autoregressive order, the number of regimes, and the dimension of the time series, respectively. The argument `params` should be used to specify the parameter values, whereas the weight function and weight function parameters are specified in the arguments `weight_function` and `weightfun_pars`, respectively, and the conditional distribution in the function `cond_dist`.

To exemplify, we build a reduced form Gaussian STVAR $p = 1$, $M = 1$, $d = 2$ model with relative stationary densities as the transition weights. The model has intercept parametrization and parameter values $\varphi_{1,0} = (0, 1)$, $\varphi_{12,0} = (0, 2)$, $\text{vec}(A_{1,1}) = (0.2, 0.2, 0.2, -0.2)$, $\text{vec}(A_{1,1}) = (0.3, 0.3, 0.3, -0.3)$, $\text{vech}(\Omega_1) = (1, 0.1, 1)$, $\text{vech}(\Omega_2) = (4, 0.4, 4)$, and α_1=0.6. After building the model, we use the `print` method to examine it:

```
R> params122 <- c(0, 1, 0, 2, 0.2, 0.2, 0.2, -0.2, 0.3, 0.3, 0.3, -0.3, 1,
+    0.1, 1, 4, 0.4, 4, 0.6)
R> mod122 <- STVAR(p=1, M=2, d=2, params=params122,
+    weight_function="relative_dens")
R> mod122

relative_dens Gaussian STVAR model, reduced form model, no AR_constraints,
no mean_constraints,
```

```
  p = 1, M = 2, d = 2, #parameters = 19,

Regime 1
Weight param: 0.60
Regime means: 0.22, 0.87

   Y     phi0          A1                   Omega        1/2
1 y1 = [ 0.00 ] + [  0.20  0.20 ] y1.1 + [  1.00 0.10 ]     eps1
2 y2   [ 1.00 ]   [  0.20 -0.20 ] y2.1   [  0.10 1.00 ]     eps2

Regime 2
Weight param: 0.40
Regime means: 0.73, 1.71

   Y     phi0          A1                   Omega        1/2
1 y1 = [ 0.00 ] + [  0.30  0.30 ] y1.1 + [  4.00 0.40 ]     eps1
2 y2   [ 2.00 ]   [  0.30 -0.30 ] y2.1   [  0.40 4.00 ]     eps2
```

It is possible to include data in the models built with `STVAR` by either providing the data in the argument `data`. When the model is supplied with data, the transition weights and other data dependent statistics are calculated for the model as well.

8. Simulation and forecasting

8.1. Simulation

sstvars implements the S3 method `simulate` for simulating observations from STVAR processes (see `?simulate.stvar`). The method requires the process to be given as a class `stvar` object, which are typically created either by estimating a model with the function `fitSTVAR` (or `fitSSTVAR`) or by specifying the parameter values by hand and building the model with the constructor function `STVAR`. 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) some regime (with the argument `init_regime`). 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 500 observations long sample from the STVAR model built in Section 7, drawing initial values from the first regime:

```
R> mysim <- simulate(mod122, nsim=500, init_regime=1, 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 transition weights in `$transition_weights`.

8.2. Simulation based forecasting

Deriving multiple-steps-ahead point predictions and prediction intervals analytically for the

STVAR models is complicated, so **sstvars** 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 transition weights, which might be of interest because the researcher can often associate statistical characteristics or economic interpretations to the regimes.

Forecasting is most conveniently done with the `predict` method (see `?predict.stvar`). The available arguments include the number of steps ahead to be predicted (`nsteps`), the number sample paths the forecast is based on (`nsim`), 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` for the point forecast.

To exemplify, the following code forecasts the two-dimensional time-series of U.S. GDP and GDP deflator growth using the logistic Student's t STVAR$(1, 2)$ model `fit12` estimated in Section 4.3. The forecast is 10 steps (quarters in this case) ahead, based on 10000 Monte Carlo repetitions with the point forecast based on the mean of those repetitions. The prediction intervals are two-sided with confidence levels 0.95 and 0.90.

```
R> mypred <- predict(fit12, nsteps=10, nsim=10000, pred_type="mean",
+    pi=c(0.95, 0.90))
```

The results can be printed with the print method using the command `print(mypred)` or plotted with the plot method using the command `plot(mypred)`.

9. Summary

Smooth transition vector autoregressive models are a valuable tool in modeling multivariate time series in which the data generating dynamics vary in time, exhibiting gradual shifts in the autoregressive coefficients or conditional covariance matrices. We described the R package **sstvars**, which accommodates STVAR models with various transition weight functions, including logistic weights (Anderson and Vahid 1998), multinomial logit weights, exponential weights (e.g., Hubrich and Teräsvirta 2013), threshold weights (Tsay 1998), and transition weights that defined as weighted relative likelihoods of the regimes corresponding to the preceding p observations (FILL IN REFERENCE). Currently, the accommodated conditional distributions include Student's t distribution and Gaussian distribution, whereas the accommodated identification methods include recursive identification and identification by heteroskedasticity. We discussed several features implemented in **sstvars** for STVAR modeling: unconstrained and constrained maximum likelihood estimation of the model parameters, impulse response analysis, residual based diagnostics, hypothesis testing, simulation, forecasting, and more. For convenience, we have collected some useful functions in **sstvars** to Table 1. For all the exported functions and their usage, see the reference manual.

Computational details

The results in this paper were obtained using R 4.3.1 and **sstvars** 1.0.0 package running on

Related to	Name	Description
Estimation	`fitSTVAR`	Estimate STVAR models.
	`fitSSTVAR`	Estimate or construct structural STVAR models.
	`alt_stvar`	Construct a STVAR model based on any estimation round.
	`iterate_more`	Run more iterations of the variable metric algorithm for a preliminary estimated STVAR model.
Estimates	`summary` (method)	Detailed printout of the model.
	`plot` (method)	Plot the series with the fitted transition weights 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 about the estimate.
Diagnostics	`Portmanteau_test`	Calculate the (adjusted) Portmanteau test.
	`diagnostic_plot`	Plot residual diagnostics (raw or standardized residuals).
Forecasting	`predict` (method)	Forecast future observations and transition weights of the process.
Simulation	`simulate` (method)	Simulate from a STVAR process.
Create model	`STVAR`	Construct a STVAR model based on given parameter values.
Hypothesis testing	`LR_test`	Calculate likelihood ratio test.
	`Wald_test`	Calculate Wald test.
	`Rao_test`	Calculate Rao's test.
Impulse response analysis	`GIRF`	Estimate generalized impulse response functions.
	`GFEVD`	Estimate generalized forecast error variance decomposition.
	`linear_IRF`	Estimate linear impulse response functions.
Other	`bound_joint_spectral_radius`	Calculate an upper bound of the joint spectral radius of the companion form AR matrices of the regimes.
	`swap_parametrization`	Swap between mean and intercept parametrizations
	`swap_W_signs`	Swap the signs of the columns of the impact matrix of model identified by heteroskedasticity.
	`reorder_W_columns`	Reorder the columns of the impact matrix of model identified by heteroskedasticity.

Table 1: Some useful functions in **sstvars** 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 `stvar` object (often generated by the function `fitSTVAR`, `fitSSTVAR` or `STVAR`).

MacBook Pro 14", 2021, with Apple M1 Pro processor, 16 Gt of unified RAM, and macOS Sonoma 14.2.1 operating system.

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 likely due to the numerical error caused by the limited precision of the float point representation.

References

Anderson H, Vahid F (1998). "Testing multiple equation systems for common nonlinear components." *Journal of Econometrics*, **84**(1), 1–36. doi:10.1016/S0304-4076(97)00076-6.

Angelini G, Bacchiocchi E, Caggiano G, Fanelli L (2019). "Uncertainty across volatility regimes." *Journal of Applied Econometrics*, **34**(3), 437–455. doi:10.1002/jae.2672.

Bacchiocchi E, Castelnuovo E, Fanelli L (2016). "Gimme a Break! Identification and Estimation of the Macroeconomic Effects of Monetary Policy Shocks in the U.S." *Melbourne Institute Working Paper No. 31/16*. doi:10.2139/ssrn.2851316.

Bacchiocchi E, Fanelli L (2015). "Identification in Structural Vector Autoregressive models with Structural Changes, with an Application to US Monetary policy." *Oxford Bulletin of Economics and Statistics*, **77**(6), 761–779. doi:10.1111/obes.12092.

Blondel V, Nesterov Y (2005). "Computationally Efficient Approximations of the Joint Spectral Radius." *SIAM Journal on Matrix Analysis and Applications*, **27**(1), 256–272. doi:10.1137/040607009.

Burgard J, Neuenkirch M, Nöckel M (2019). "State-Dependent Transmission of Monetary Policy in the Euro Area." *Journal of Money, Credit and Banking*, **51**(7), 2053–2070. doi:10.1111/jmcb.12592.

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.

Chang CT, Blondel V (2013). "An experimental study of approximation algorithms for the joint spectral radius." *Numerical Algorithms*, **64**, 181–202. doi:10.1007/s11075-012-9661-z.

Herwartz H, Lütkepohl H (2014). "Structural vector autoregressions with Markov switching: Combining conventional with statistical identification of shocks." *Journal of Econometrics*, **183**(1), 104–116. doi:doi.org/10.1016/j.jeconom.2014.06.012.

Hubrich K, Teräsvirta T (2013). "Thresholds and Smooth Transitions in Vector Autoregressive Models." *CREATES Research Paper 2013-18, Aarhus University*. URL https://pure.au.dk/ws/files/54473123/rp13_18.pdf.

Jungers R (2023). *The JSR toolbox (https://www.mathworks.com/matlabcentral/fileexchange/33202-the-jsr-toolbox), MATLAB Central File Exchange*.

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.

Kheifets I, Saikkonen P (2020). "Stationarity and ergodicity of vector STAR models." *Econometric Reviews*, **39**(407–414), 1311–1324. doi:10.1080/07474938.2019.1651489.

Kilian L, Lütkepohl H (2017). *Structural Vector Autoregressive Analysis*. 1st edition. Cambridge University Press, Cambridge. doi:10.1017/9781108164818.

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, Nyberg H (2016). "Generalized Forecast Error Variance Decomposition for Linear and Nonlinear Multivariate Models." *Oxford Bulletin of Economics and Statistics*, **78**(4), 595–603. doi:10.1111/obes.12125.

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

Lütkepohl H, Netšunajev A (2017). "Structural vector autoregressions with smooth transitions in variances." *Journal of Economic Dynamics & Control*, **84**, 43–57. doi:10.1016/j.jedc.2017.09.001.

Meitz M, Preve D, Saikkonen P (2023). "A mixture autoregressive model based on Student's t-distribution." *Communications in Statistics - Theory and Methods*, **52**(2), 499–515. doi:10.1080/03610926.2021.1916531.

Muirhead R (1982). *Aspects of Multivariate Statistical Theory*. 1st edition. John Wiley & Sons, Hoboken, New Jersey. doi:10.1002/9780470316559.

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.

Parrilo P, Jadbabaie A (2008). "Approximation of the joint spectral radius using sum of squares." *Linear Algebra and its Applications*, **428**(10), 2385–2402. doi:10.1016/j.laa.2007.12.027.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rigobon R (2003). "Identification through Heteroskedasticity." *The Review of Economics and Statistics*, **85**(4), 777–792. doi:10.1162/003465303772815727.

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.

Teräsvirta T (1994). "Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models." *Journal of the American Statistical Association*, **89**(425), 208–2187. doi:10.2307/2291217.

Tsay R (1998). "Testing and Modeling Multivariate Threshold Models." *Journal of the American Statistical Association*, **93**(443), 1188–1202. doi:10.1080/01621459.1998.10473779.

Virolainen S (2018a). *gmvarkit: Estimate Gaussian and Student's t Mixture Vector Autoregressive Models*. R package version 2.1.0 available at CRAN: https://CRAN.R-project.org/package=gmvarkit, URL https://CRAN.R-project.org/package=gmvarkit.

Virolainen S (2018b). *uGMAR: Estimate Univariate Gaussian and Student's t Mixture Autoregressive Models*. R package version 3.4.2 available at CRAN: https://CRAN.R-project.org/package=uGMAR, URL https://CRAN.R-project.org/package=uGMAR.

Virolainen S (2021). "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.

Virolainen S (2022a). *Essays on mixture autoregressive models with applications to macroeconomics and finance*. Ph.D. thesis, University of Helsinki. URL http://hdl.handle.net/10138/350319.

Virolainen S (2022b). "A mixture autoregressive model based on Gaussian and Student's t-distributions." *Studies in Nonlinear Dynamics & Econometrics*, **26**(4), 559–580. 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