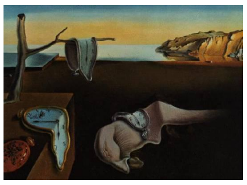

Journal of Statistical Software

November 2021, Volume 100, Issue 13. *doi: 10.18637/jss.v100.i13*

Shrinkage in the Time-Varying Parameter Model Framework Using the R Package shrinkTVP

Peter Knaus
WU Vienna

Angela Bitto-Nemling
JKU Linz

Annalisa Cadonna
Crayon Austria

Sylvia Frühwirth-Schnatter
WU Vienna

Abstract

Time-varying parameter (TVP) models are widely used in time series analysis to flexibly deal with processes which gradually change over time. However, the risk of overfitting in TVP models is well known. This issue can be dealt with using appropriate global-local shrinkage priors, which pull time-varying parameters towards static ones. In this paper, we introduce the R package **shrinkTVP** (Knaus, Bitto-Nemling, Cadonna, and Frühwirth-Schnatter 2021), which provides a fully Bayesian implementation of shrinkage priors for TVP models, taking advantage of recent developments in the literature, in particular those of Bitto and Frühwirth-Schnatter (2019) and Cadonna, Frühwirth-Schnatter, and Knaus (2020). The package **shrinkTVP** allows for posterior simulation of the parameters through an efficient Markov Chain Monte Carlo scheme. Moreover, summary and visualization methods, as well as the possibility of assessing predictive performance through log-predictive density scores, are provided. The computationally intensive tasks have been implemented in C++ and interfaced with R. The paper includes a brief overview of the models and shrinkage priors implemented in the package. Furthermore, core functionalities are illustrated, both with simulated and real data.

Keywords: Bayesian inference, Gibbs sampler, Markov chain Monte Carlo (MCMC), normal-gamma prior, time-varying parameter (TVP) models, log-predictive density scores.

1. Introduction

Time-varying parameter (TVP) models are widely used in time series analysis, because of their flexibility and ability to capture gradual changes in the model parameters over time.

The popularity of TVP models in macroeconomics and finance is based on the fact that, in most applications, the influence of certain predictors on the outcome variables varies over time (Primiceri 2005; Dangl and Halling 2012; Belmonte, Koop, and Korobolis 2014). TVP models, while capable of reproducing salient features of the data in a very effective way, present a concrete risk of overfitting, as often only a small subset of the parameters are time-varying. Hence, in the last decade, there has been a growing need for models and methods able to discriminate between time-varying and static parameters in TVP models. A key contribution in this direction was the introduction of the non-centered parameterization of TVP models in Frühwirth-Schnatter and Wagner (2010), which recasts the problem of variance selection and shrinkage in terms of variable selection, thus allowing any tool used to this end in multiple regression models to be used to perform selection or shrinkage of variances. Frühwirth-Schnatter and Wagner (2010) employ a spike and slab prior, while continuous shrinkage priors have been utilised as a regularization alternative in, e.g., Belmonte *et al.* (2014), Bitto and Frühwirth-Schnatter (2019) and Cadonna *et al.* (2020). For an excellent review of shrinkage priors, with a particular focus on high dimensional regression, the reader is directed to Bhadra, Datta, Polson, and Willard (2019).

In this paper, we describe the R package **shrinkTVP** (Knaus *et al.* 2021) for Bayesian TVP models with shrinkage. The package is available under the General Public License (GPL $\geq$ 2) from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=shrinkTVP`. The package efficiently implements recent developments in the Bayesian literature, in particular the ones presented in Bitto and Frühwirth-Schnatter (2019) and Cadonna *et al.* (2020). The computationally intensive Markov chain Monte Carlo (MCMC) algorithms in the package are written in C++ and interfaced with R (R Core Team 2021) via the **Rcpp** (Eddelbuettel and François 2011; Eddelbuettel and Balamuta 2018) and the **RcppArmadillo** (Eddelbuettel and Sanderson 2014) packages. This approach combines the ease-of-use of R and its underlying functional programming paradigm with the computational speed of C++.

The package **shrinkTVP** is designed to provide an easy entry point for fitting TVP models with shrinkage priors, while also giving more experienced users the option to adapt the model to their needs. This is achieved by providing a robust baseline model that can be estimated by only passing the data, while also allowing the user to specify more advanced options. Additionally, the **shrinkTVP** package is designed to ensure compatibility with well-known times series formats and to complement other packages. As input objects, time series from the R packages **zoo** (Zeileis and Grothendieck 2005; Zeileis, Grothendieck, and Ryan 2021) and **xts** (Ryan and Ulrich 2020) as well as time series formats like `ts` are supported. Estimation output is compatible with the popular R package **coda** (Plummer, Best, Cowles, and Vines 2006; Plummer *et al.* 2020) which can be easily applied for convergence diagnostic tests, among others. Coupled with intuitive summary and plot methods, **shrinkTVP** is a package that is easy to use while remaining highly flexible.

shrinkTVP is, to our knowledge, the only R package that combines TVP models with shrinkage priors on the time-varying components in a Bayesian framework. Several R packages deal with statistical inference for various specific classes of state space models, of which TVP models are a special case. The most popular R package in this field is **dlm** (Petris 2010), a comprehensive package providing routines for maximum likelihood estimation, Kalman filtering and smoothing, and Bayesian analysis for dynamic linear models (DLMs). The accompanying book (Petris, Petrone, and Campagnoli 2009) introduces the methodology and many R code examples. As of now, priors are not designed to encourage shrinkage and **shrinkTVP** complements **dlm** in this regard.

The R package **bvarsv** (Krüger 2015) implements Bayesian inference for vector autoregressive (VAR) models with time-varying parameters (TVP-VAR) and stochastic volatility for multivariate time series as introduced by (Primiceri 2005). We refer to (Del Negro and Primiceri 2015) for details on the MCMC algorithm and a later correction of the original scheme. Kalman filter type algorithm, In addition to the very user friendly estimation function `bvar.sv.tvp()`, **bvarsv** provides posterior predictive distributions and enables impulse response analysis. The package includes the macroeconomic data set analysed in (Primiceri 2005) as example data set, `usmacro.update`, which we use in our predictive exercise in Section 5 to showcast the effect of introducing shrinkage priors on time-varying parameters.

Additional packages emerged very recently. The R package **tvReg** (Casas and Fernandez-Casal 2021) presents a user friendly compendium of many common linear TVP models, including standard linear regression as well as autoregressive, seemingly unrelated equation and VAR models. Estimation is based on kernel smoothing techniques. For an illustrative application, a TVP-VAR(4) model is fitted to the `usmacro.update` data set mentioned above, using the function `tvVAR()`. The R package **walker** (Helske 2021) facilitates the estimation of DLMs and generalized DLMs using MCMC algorithms provided by Stan (Carpenter *et al.* 2017). For inference, the importance sampling method of (Vihola, Helske, and Franks 2017) is implemented within a Hamiltonian Monte Carlo framework. The R package **bsts** (Scott 2021) performs Bayesian analysis for structural time series models, a highly relevant class of state space models including DLMs. **bsts** is a very powerful package that allows shrinkage for static regression coefficients using spike and slab priors. However, as for any other packages mentioned above, variation of the dynamic components is not regularized in **bsts**.

A main contribution of the package **shrinkTVP** is bridging the active field of R packages for state space models with the even more active field of R packages that provide regularization and shrinkage methods for common regression type models.

Among others, **ncvreg** (Breheny and Huang 2011) is useful for fitting standard penalized regression estimators, **glmnet** (Friedman, Hastie, and Tibshirani 2010) allows elastic-net regularization for a variety of models, **horseshoe** (Van der Pas, Scott, Chakraborty, and Bhattacharya 2019) implements the horseshoe prior, while **shrink** (Dunkler, Sauerbrei, and Heinze 2016) provides various shrinkage methods for linear, generalized linear, and Cox regression models. **biglasso** (Zeng and Breheny 2017) aims at very fast lasso-type regularization for high-dimensional linear regression. Recent R packages include **NormalBetaPrime** (Bai and Ghosh 2019) for Bayesian univariate and **MBSP** (Bai and Ghosh 2018) for Bayesian multivariate linear regression analysis using, respectively, the normal-beta prime and the three parameter beta normal family for inducing shrinkage. The R package **monomvn** (Gramacy 2019) employs a normal-gamma prior in the specific situation of Bayesian inference for multivariate normal and Student-t data with a monotone pattern of missing data.

The remainder of the paper is organized as follows. Section 2 briefly introduces TVP models and normal-gamma-gamma shrinkage priors, and describes the MCMC algorithms for posterior simulation. The package **shrinkTVP** is introduced in Section 3. In particular, we illustrate how to run the MCMC sampler using the main function `shrinkTVP()`, how to choose a specific model, and how to conduct posterior inference using the returned object of class '`shrinkTVP`'. Section 4 explains how to assess model performance by calculating log-predictive density scores (LPDSs), and how to use LPDSs to compare the predictive performances of different priors. This is illustrated using the `usmacro.update` data set. Finally, Section 6 concludes the paper.

2. Model specification and estimation

2.1. TVP models

Let us recall the state space form of a TVP model. For $t = 1, \dots, T$, we have that

$$\begin{aligned} y_t &= \boldsymbol{x}_t \boldsymbol{\beta_t} + \epsilon_t, & \epsilon_t &\sim \mathcal{N}(0, \sigma_t^2), \\ \boldsymbol{\beta}_t &= \boldsymbol{\beta}_{t-1} + \boldsymbol{w}_t, & \boldsymbol{w}_t &\sim \mathcal{N}_d(0, \mathbf{Q}), \end{aligned} \tag{1}$$

where y_t is a univariate response variable and $\boldsymbol{x}_t = (x_{t1}, x_{t2}, \dots, x_{td})$ is a d-dimensional row vector containing the regressors at time t, with x_{t1} corresponding to the intercept. For simplicity, we assume here that $\mathbf{Q} = \text{Diag}(\theta_1, \dots, \theta_d)$ is a diagonal matrix, implying that the state innovations are conditionally independent. Moreover, we assume the initial value follows a normal distribution, i.e., $\boldsymbol{\beta}_0 \sim \mathcal{N}_d(\boldsymbol{\beta}, \mathbf{Q})$, with initial mean $\boldsymbol{\beta} = (\beta_1, \dots, \beta_d)$. Model 1 can be rewritten equivalently in the non-centered parametrization as

$$\begin{aligned} y_t &= \boldsymbol{x}_t \boldsymbol{\beta} + \boldsymbol{x}_t \text{Diag}(\sqrt{\theta}_1, \dots, \sqrt{\theta}_d) \tilde{\boldsymbol{\beta}}_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_t^2), \\ \tilde{\boldsymbol{\beta}}_t &= \tilde{\boldsymbol{\beta}}_{t-1} + \tilde{\boldsymbol{u}}_t, \qquad \tilde{\boldsymbol{u}}_t \sim \mathcal{N}_d(0, I_d), \end{aligned} \tag{2}$$

with $\tilde{\boldsymbol{\beta}}_0 \sim \mathcal{N}_d(\mathbf{0}, I_d)$, where I_d is the d-dimensional identity matrix.

shrinkTVP is capable of modelling the observation error both homoscedastically, i.e., $\sigma_t^2 \equiv \sigma^2$ for all $t = 1, \dots, T$ and heteroscedastically, via a stochastic volatility (SV) specification. In the latter case, the log-volatility $h_t = \log \sigma_t^2$ follows an AR(1) model (Jacquier, Polson, and Rossi 1994; Kastner and Frühwirth-Schnatter 2014; Kastner 2016). More specifically,

$$h_t \mid h_{t-1}, \mu, \phi, \sigma_\eta^2 \sim \mathcal{N}\left(\mu + \phi(h_{t-1} - \mu), \sigma_\eta^2\right), \tag{3}$$

with initial state $h_0 \sim \mathcal{N}\left(\mu, \sigma_\eta^2/(1 - \phi^2)\right)$. The stochastic volatility model on the errors can prevent the detection of spurious variations in the TVP coefficients (Nakajima 2011; Sims 2001) by capturing some of the variability in the error term.

2.2. Prior specification

Shrinkage priors on variances and model parameters

We place conditionally independent normal-gamma-gamma (NGG) priors (Cadonna *et al.* 2020; Griffin and Brown 2017), both on the standard deviations of the innovations, that is the $\sqrt{\theta_j}$'s, and on the means of the initial value β_j, for $j = 1, \dots, d$. Note that, in the case of the standard deviations, this can equivalently be seen as a triple gamma prior on the innovation variances θ_j, for $j = 1, \dots, d$. The NGG can be represented as a conditionally normal distribution, where the component specific variance is itself a compound probability distribution resulting from two gamma distributions. In this representation, it looks as follows

$$\sqrt{\theta}_j \mid \xi_j^2 \sim \mathcal{N}\left(0, \xi_j^2\right), \qquad \xi_j^2 \mid a^\xi, \kappa_j^2 \sim \mathcal{G}\left(a^\xi, \frac{a^\xi \kappa_j^2}{2}\right), \quad \kappa_j^2 \mid c^\xi, \kappa_B^2 \sim \mathcal{G}\left(c^\xi, \frac{c^\xi}{\kappa_B^2}\right) \tag{4}$$

$$\beta_j \mid \tau_j^2 \sim \mathcal{N}\left(0, \tau_j^2\right), \qquad \tau_j^2 \mid a^\tau, \lambda_j^2 \sim \mathcal{G}\left(a^\tau, \frac{a^\tau \lambda_j^2}{2}\right) \quad \lambda_j^2 \mid c^\tau, \lambda_B^2 \sim \mathcal{G}\left(c^\tau, \frac{c^\tau}{\lambda_B^2}\right). \tag{5}$$

Letting c^ξ and c^τ go to infinity results in a normal-gamma (NG) prior (Griffin and Brown 2010) on the $\sqrt{\theta}_j$'s and β_j's. It has a representation as a conditionally normal distribution, with the component specific variance following a gamma distribution, that is

$$\sqrt{\theta}_j \mid \xi_j^2 \sim \mathcal{N}\left(0, \xi_j^2\right), \qquad \xi_j^2 \mid a^\xi, \kappa_B^2 \sim \mathcal{G}\left(a^\xi, \frac{a^\xi \kappa_B^2}{2}\right), \tag{6}$$

$$\beta_j \mid \tau_j^2 \sim \mathcal{N}\left(0, \tau_j^2\right), \qquad \tau_j^2 \mid a^\tau, \lambda_B^2 \sim \mathcal{G}\left(a^\tau, \frac{a^\tau \lambda_B^2}{2}\right). \tag{7}$$

From here, letting a^ξ and a^τ go to infinity yields a normal prior with fixed variance, also known as ridge regression:

$$\sqrt{\theta}_j \mid \kappa_B^2 \sim \mathcal{N}\left(0, \frac{2}{\kappa_B^2}\right), \tag{8}$$

$$\beta_j \mid \lambda_B^2 \sim \mathcal{N}\left(0, \frac{2}{\lambda_B^2}\right). \tag{9}$$

We refer to a^ξ and a^τ as the pole parameters, as marginally more mass is placed around zero as they become smaller. c^ξ and c^τ are referred to as the tail parameters, as they control the amount of mass in the tails of the distribution, with smaller values equating to heavier tails. Finally, the parameters κ_B^2 and λ_B^2 are dubbed the global shrinkage parameters, as they influence how strongly all parameters are pulled to zero. The larger κ_B^2 and λ_B^2, the stronger this effect.

One of the key benefits of the NGG prior is that many interesting shrinkage priors are contained within it as special or limiting cases. Beyond the NG prior mentioned above, two such cases are the horseshoe prior (Carvalho, Polson, and Scott 2009) and the Bayesian Lasso (Park and Casella 2008). The former results from an NGG prior with the pole and tail parameters equal to 0.5, while the latter is a special case of the NG prior with a pole parameter fixed to one. As the connection between the NGG prior and the horseshoe prior may not be entirely obvious from the parameterization presented here, the interested reader is referred to Cadonna *et al.* (2020) for details.

The parameters a^ξ, a^τ, c^ξ, c^τ, κ_B^2 and λ_B^2 can be learned from the data through appropriate prior distributions. Results from Cadonna *et al.* (2020) motivate the use of different distributions for these parameters under the NGG and NG prior. In the NGG case, the scaled global shrinkage parameters conditionally follow F distributions, depending on their respective pole and tail parameters:

$$\frac{\kappa_B^2}{2} \mid a^\xi, c^\xi \sim F(2a^\xi, 2c^\xi), \qquad \frac{\lambda_B^2}{2} \mid a^\tau, c^\tau \sim F(2a^\tau, 2c^\tau). \tag{10}$$

The scaled tail and pole parameters, in turn, follow beta distributions:

$$2a^{\xi} \sim \mathcal{B}\left(\alpha_{a^{\xi}}, \beta_{a^{\xi}}\right), \qquad 2c^{\xi} \sim \mathcal{B}\left(\alpha_{c^{\xi}}, \beta_{c^{\xi}}\right), \tag{11}$$

$$2a^{\tau} \sim \mathcal{B}\left(\alpha_{a^{\tau}}, \beta_{a^{\tau}}\right), \qquad 2c^{\tau} \sim \mathcal{B}\left(\alpha_{c^{\tau}}, \beta_{c^{\tau}}\right). \tag{12}$$

These priors are chosen as they imply a uniform prior on a suitably defined model size, see Cadonna *et al.* (2020) for details. In the NG case the global shrinkage parameters follow independent gamma distributions:

$$\kappa_B^2 \sim \mathcal{G}(d_1, d_2), \qquad \lambda_B^2 \sim \mathcal{G}(e_1, e_2). \tag{13}$$

In order to learn the pole parameters in the NG case, we generalize the approach taken in Bitto and Frühwirth-Schnatter (2019) and place the following gamma distributions as priors:

$$a^{\xi} \sim \mathcal{G}(\alpha_{a^{\xi}}, \alpha_{a^{\xi}}\beta_{a^{\xi}}), \qquad a^{\tau} \sim \mathcal{G}(\alpha_{a^{\tau}}, \alpha_{a^{\tau}}\beta_{a^{\tau}}), \tag{14}$$

which correspond to the exponential priors used in Bitto and Frühwirth-Schnatter (2019) when $\alpha_{a^{\xi}} = 1$ and $\alpha_{a^{\tau}} = 1$. The parameters $\alpha_{a^{\xi}}$ and $\alpha_{a^{\tau}}$ act as degrees of freedom and allow the prior to be bounded away from zero.

Prior on the volatility parameter

In the homoscedastic case we employ a hierarchical prior, where the scale of an inverse gamma prior for σ^2 follows a gamma distribution, that is,

$$\sigma^2 \mid C_0 \sim \mathcal{G}^{-1}\left(c_0, C_0\right), \qquad C_0 \sim \mathcal{G}\left(g_0, G_0\right), \tag{15}$$

with hyperparameters c_0, g_0, and G_0.

In the case of stochastic volatility, the priors on the parameters μ, ϕ and σ_η^2 in Equation 3 are chosen as in Kastner and Frühwirth-Schnatter (2014), that is

$$\mu \sim \mathcal{N}(b_\mu, B_\mu), \qquad \frac{\phi + 1}{2} \sim \mathcal{B}(a_\phi, b_\phi), \qquad \sigma_\eta^2 \sim \mathcal{G}(1/2, 1/2B_\sigma), \tag{16}$$

with hyperparameters $b_\mu, B_\mu, a_\phi, b_\phi$, and B_σ.

2.3. MCMC sampling algorithm

The package **shrinkTVP** implements an MCMC Gibbs sampling algorithm with Metropolis-Hastings steps to obtain draws from the posterior distribution of the model parameters. Here, we roughly sketch the sampling algorithm and refer the interested reader to Bitto and Frühwirth-Schnatter (2019) and Cadonna *et al.* (2020) for further details.

Step 4 presents a fork in the algorithm, as different parameterizations are used in the NGG and NG case, as to improve mixing. For details on the exact parameterization used in the NGG case, see Cadonna *et al.* (2020). Additionally, not all sampling steps are performed in all prior setups. If, for example, the user has defined that κ_B^2 should not be learned from the data, then this step is not executed.

One key feature of the algorithm is the joint sampling of the time-varying parameters $\tilde{\boldsymbol{\beta}}_t$, for $t = 0, \ldots, T$ in step 1 of Algorithm 1. We employ the procedure described in McCausland, Miller, and Pelletier (2011) which exploits the sparse, block tri-diagonal structure of

Algorithm 1 Gibbs Sampling Algorithm

1. Sample the latent states $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_0, \dots, \tilde{\boldsymbol{\beta}}_T)$ in the non-centered parametrization from a multivariate normal distribution.
2. Sample jointly $\beta_1, \dots, \beta_d$, and $\sqrt{\theta_1}, \dots, \sqrt{\theta_d}$ in the non-centered parametrization from a multivariate normal distribution.
3. Perform an ancillarity-sufficiency interweaving step and redraw each $\beta_1, \dots, \beta_d$ from a normal distribution and each $\theta_1, \dots, \theta_d$ from a generalized inverse Gaussian distribution using **GIGrvg** (Leydold and Hörmann 2017).
4. Sample the prior variances $\xi_1^2, \dots \xi_d^2$ and $\tau_1^2, \dots \tau_d^2$ and the component specific hyperparameters. Sample (where required) the pole, tail and global shrinkage parameters. In the NGG case, this is done by emplyoing steps (c) - (f) from Algorithm 1 in Cadonna *et al.* (2020). In the NG case steps (d) and (e) from Algorithm 1 in Bitto and Frühwirth-Schnatter (2019) are used. In the ridge regression case simply $\xi_j^2 = 2/\kappa_B^2$ and $\tau_j^2 = 2/\lambda_B^2$, for $d = 1, \dots, d$.
5. Sample the error variance σ^2 from an inverse gamma distribution in the homoscedastic case or, in the SV case, sample the level μ, the persistence ϕ, the volatility of the volatility σ_η^2 and the log-volatilities $\boldsymbol{h} = (h_0, \dots, h_T)$ using **stochvol** (Kastner 2016; Hosszejni and Kastner 2021).

the precision matrix of the full conditional distribution of $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_0, \dots, \tilde{\boldsymbol{\beta}}_T)$, to speed up computations.

Moreover, as described in Bitto and Frühwirth-Schnatter (2019), in step 3 we make use of the ancillarity-sufficiency interweaving strategy (ASIS) introduced by Yu and Meng (2011). ASIS is well known to improve mixing by sampling certain parameters both in the centered and non-centered parameterization. This strategy has been successfully applied to univariate SV models (Kastner and Frühwirth-Schnatter 2014), multivariate factor SV models (Kastner, Frühwirth-Schnatter, and Lopes 2017) and dynamic linear state space models (Simpson, Niemi, and Roy 2017).

Adaptive Metropolis-within-Gibbs For the pole and tail parameters, no full conditionals exist and a Metropolis-Hastings step has to be performed. To improve mixing, `shrinkTVP` supports adaptive Metropolis-within-Gibbs as in Roberts and Rosenthal (2009). The algorithm works as follows. For each parameter i that is being learned from the data, let s_i represent the standard deviation of the proposal distribution. After the n_i^{th} batch of m_i iterations, update s_i according to the following rule:

- increase the log of s_i by $\min(c_i, n_i^{1/2})$ if the acceptance rate of the previous batch was above d_i or
- decrease the log of s_i by $\min(c_i, n_i^{1/2})$ if the acceptance rate of the previous batch was below d_i.

The starting value of s_i, m_i, c_i and d_i can all be set by the user. Additionally, if adaptive Metropolis-within-Gibbs is not desired, it can be switched off and a simple Metropolis-Hastings step will be performed.

3. The shrinkTVP package

3.1. Running the model

The core function of the package **shrinkTVP** is the function `shrinkTVP()`, which serves as an R-wrapper for the actual sampler coded in C++. The function works out-of-the-box, meaning that estimation can be performed with minimal user input. With default settings, the TVP model in Equation 1 is estimated in a Bayesian fashion with the NG prior defined in Equation (6), Equation (7), Equation (13) and Equation (14) with the following choice for the hyperparameters: $d_1 = d_2 = e_1 = e_2 = 0.001$, $\alpha_{a^\xi} = \alpha_{a^\tau} = 5$ and $\beta_{a^\xi} = \beta_{a^\tau} = 10$, implying a prior mean of $\mathrm{E}(a^\xi) = \mathrm{E}(a^\tau) = 0.1$. The error is assumed to be homoscedastic, with prior defined in Equation (15) and hyperparameters $c_0 = 2.5$, $g_0 = 5$, and $G_0 = g_0/(c_0 - 1)$.

The only compulsory argument is an object of class '`formula`', which most users will be familiar with (see, for example, the use in the function `lm()` in the package **stats** (R Core Team 2021)). The second argument is an optional data frame, containing the response variable and the covariates. Exemplary usage of this function is given in the code snippet below, along with the default output. All code was on run on a personal computer with an Intel i5-8350U CPU.

```
R> library("shrinkTVP")
R> set.seed(123)
R> sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
R> data <- sim$data
R> res <- shrinkTVP(y ~ x1 + x2, data = data)

0%   10   20   30   40   50   60   70   80   90   100%
[----|----|----|----|----|----|----|----|----|----|
**************************************************|
Timing (elapsed): 3.403 seconds.
4408 iterations per second.

Converting results to coda objects and summarizing draws... Done!
```

Note that the data in the example is generated by the function `simTVP()`, which can create synthetic datasets of varying sizes for illustrative purposes. The inputs `theta` and `beta` can be used to specify the true $\theta_1, \ldots, \theta_d$ and $\beta_1, \ldots, \beta_d$ used in the data generating process, in order to evaluate how well `shrinkTVP` recaptures these true values. The values correspond to the ones used in the synthetic example of Bitto and Frühwirth-Schnatter (2019).

The user can specify the following MCMC algorithm parameters: `niter`, which determines the number of MCMC iterations including the burn-in, `nburn`, which equals the number of MCMC iterations discarded as burn-in, and `nthin`, indicating the thinning parameter, meaning that every nthin-th draw is kept and returned. The default values are `niter = 10000`, `nburn = round(niter/2)` and `nthin = 1`.

The user is strongly encouraged to check convergence of the produced Markov chain, especially for a large number of covariates. The output is made **coda** compatible, so that the user can utilize the tools provided by the excellent R package to assess convergence.

	Shrinkage on $\sqrt{\theta_j}$			Shrinkage on β_j		
	c^ξ	a^ξ	κ_B^2	c^τ	a^τ	λ_B^2
NGG prior						
Fully hierarchical NGG	$\mathcal{B}(\alpha_{c^\xi}, \beta_{c^\xi})$	$\mathcal{B}(\alpha_{a^\xi}, \beta_{a^\xi})$	$F(2a^\xi, 2c^\xi)$	$\mathcal{B}(\alpha_{c^\tau}, \beta_{c^\tau})$	$\mathcal{B}(\alpha_{a^\tau}, \beta_{a^\tau})$	$F(2a^\tau, 2c^\tau)$
Hierarchical NGG	fixed	fixed	$F(2a^\xi, 2c^\xi)$	fixed	fixed	$F(2a^\tau, 2c^\tau)$
NGG	fixed	fixed	fixed	fixed	fixed	fixed
Hierarchical Horseshoe	fixed at 0.5	fixed at 0.5	$F(2a^\xi, 2c^\xi)$	fixed at 0.5	fixed at 0.5	$F(2a^\tau, 2c^\tau)$
Horseshoe	fixed at 0.5	fixed at 0.5	fixed	fixed at 0.5	fixed at 0.5	fixed
NG prior						
Fully hierarchical NG	-	$\mathcal{G}(\alpha_{a^\xi}, \alpha_{a^\xi}\beta_{a^\xi})$	$\mathcal{G}(d_1, d_2)$	-	$\mathcal{G}(\alpha_{a^\tau}, \alpha_{a^\tau}\beta_{a^\tau})$	$\mathcal{G}(e_1, e_2)$
Hierarchical NG	-	fixed	$\mathcal{G}(d_1, d_2)$	-	fixed	$\mathcal{G}(e_1, e_2)$
NG	-	fixed	fixed	-	fixed	fixed
Bayesian Lasso	-	fixed at 1	fixed	-	fixed at 1	fixed
Ridge regression	-	-	fixed	-	-	fixed

Table 1: Overview of different possible model specifications. Note that in the NGG prior case, the priors on the hyperparameters are scaled (e.g., $2a^\xi \sim \mathcal{B}(\alpha_{a^\xi}, \beta_{a^\xi})$). These scalings are omitted from this table for the sake of brevity. See Section 2.2 for details.

3.2. Specifying the priors

More granular control over the prior setup can be exercised by passing additional arguments to `shrinkTVP`. The most important argument in this regard is `mod_type`, which is used to specify whether the normal-gamma-gamma (`mod_type = "triple"`), the normal-gamma (`mod_type = "double"`) or ridge regression (`mod_type = "ridge"`) is used. Beyond this, the user can specify the hyperparameters given in Section 2.2 and has the possibility to fix one or both of the values of the global shrinkage parameters (κ_B^2, λ_B^2) and the pole and tail parameters (a^τ, a^ξ, c^τ, c^ξ). By default, these parameters are learned from the data. The benefit of this flexibility is twofold: on the one hand, desired degrees of sparsity and global shrinkage can be achieved through fixing the hyperparameters; on the other hand, interesting special cases arise from setting certain values of hyperparameters. Under an NGG prior, for example, setting the pole and tail parameters equal to 1/2 results in a horseshoe prior on the $\sqrt{\theta_j}$'s and the β_j's, respectively. If the user desires a higher degree of sparsity, this can be achieved by setting the pole parameters to a value closer to zero. Table 1 gives an overview of different model specifications. Note that different hyperparameter values can be chosen for the variances and the means of the initial values.

In the following, we give some examples of models that can be estimated with the **shrinkTVP** package. In particular, we demonstrate how certain combinations of input arguments correspond to different model specifications. If the learning of a parameter is deactivated and no specific fixed value is provided, `shrinkTVP` will resort to default values. These equate to 0.1 for the pole and tail parameters and 20 for the global shrinkage parameters. Note that in the following snippets of code, the argument `display_progress` is always set to `FALSE`, in order to suppress the progress bar and other outputs.

Fixing the pole parameters It is possible to set the pole parameter $a^\xi (a^\tau)$ to a fixed value through the input argument `a_xi` (`a_tau`), after setting `learn_a_xi` (`learn_a_tau`) to

FALSE. As an example, we show how to fit a hierarchical Bayesian Lasso, both on the $\sqrt{\theta_j}$'s and on the β_j's:

```
R> res_hierlasso <- shrinkTVP(y ~ x1 + x2, data = data,
+    learn_a_xi = FALSE,  learn_a_tau = FALSE,
+    a_xi = 1, a_tau = 1, display_progress = FALSE)
```

Fixing the global shrinkage parameters The user can choose to fix the value of $\kappa_B^2 (\lambda_B^2)$ by specifying the argument kappa2_B (lambda2_B), after setting learn_kappa2_B (learn_lambda2_B) to FALSE. In the code below, we give an example on how to fit a (non-hierarchical) Bayesian Lasso on both $\sqrt{\theta_j}$'s and β_j's, with corresponding global shrinkage parameters fixed both to 100:

```
R> res_lasso <- shrinkTVP(y ~ x1 + x2, data = data,
+    learn_a_xi = FALSE, learn_a_tau = FALSE, a_xi = 1, a_tau = 1,
+    learn_kappa2_B = FALSE, learn_lambda2_B = FALSE,
+    kappa2_B = 100, lambda2_B = 100,
+    display_progress = FALSE)
```

Changing the prior type To change the model type, the input argument mod_type has to be supplied. It has to be a string equal to either "triple", "double" or "ridge". As an example, we fit a hierarchical NGG prior, both on the $\sqrt{\theta_j}$'s and on the β_j's:

```
R> res_tg <- shrinkTVP(y ~ x1 + x2, data = data,
+    mod_type = "triple", display_progress = FALSE)
```

Fixing the tail parameters Much like the pole parameters, the tail parameter c^ξ (c^τ) can also be fixed to a value. This is done by setting learn_c_xi (learn_c_tau) to FALSE and then supplying the input parameter c_xi (c_tau). As an example, the code below fits a non-hierarchical horseshoe prior, both on the $\sqrt{\theta_j}$'s and on the β_j's:

```
R> res_hs <- shrinkTVP(y ~ x1 + x2, data = data,
+    mod_type = "triple",
+    learn_a_xi = FALSE, learn_a_tau = FALSE, a_xi = 0.5, a_tau = 0.5,
+    learn_c_xi = FALSE, learn_c_tau = FALSE, c_xi = 0.5, c_tau = 0.5,
+    learn_kappa2_B = FALSE, learn_lambda2_B = FALSE,
+    display_progress = FALSE)
```

3.3. Stochastic volatility specification

The stochastic volatility specification defined in Equation (3) can be used by setting the option sv to TRUE. This is made possible by a call to the update_sv() function exposed by the **stochvol** package. The code below fits a model with an NG prior in which all the parameters are learned and the observation equation errors are modeled through stochastic volatility:

```
R> res_sv <- shrinkTVP(y ~ x1 + x2, data = data, sv = TRUE,
+    display_progress = FALSE)
```

The priors on the SV parameters are the ones defined in Equation (16), with hyperparameters fixed to $b_\mu = 0$, $B_\mu = 1$, $a_\phi = 5$, $b_\phi = 1.5$, and $B_\sigma = 1$.

3.4. Specifying the hyperparameters

Beyond simply switching off parts of the hierarchical structure of the prior setup, users can also modify the hyperparameters governing the hyperprior distributions. This can be done through the arguments `hyperprior_param` and `sv_param`, which both have to be named lists. Hyperparameters not specified by the user will be set to default values, which can be found in the help file of the `shrinkTVP()` function. Note, however, that the dependence structure (e.g., κ_B^2 depends on a^ξ and c^ξ in the NGG specification) can not be changed. As such, if the user desires to change the hyperparameters of a prior that depends on other parameters, this can only be achieved by deactivating the learning of the parameters higher up in the hierarchy and fixing them to specific values. To demonstrate how to change specific hyperparameters, the code below modifies those governing the prior on a^ξ:

```
R> res_hyp <- shrinkTVP(y ~ x1 + x2, data = data,
+    hyperprior_param = list(beta_a_xi = 5, alpha_a_xi = 10),
+    display_progress = FALSE)
```

3.5. Tuning the Metropolis-Hastings steps

The Metropolis-Hastings algorithm discussed in Section 2.3 can be tuned via the argument `MH_tuning`. Similar to `hyperprior_param` and `sv_param`, it is a named list where values that are not supplied are replaced by standard values. By default, adaptive Metropolis-within-Gibbs is activated for all parameters learned from the data that requrire a Metropolis-Hastings step. Below is an example where the adaptive Metropolis is deactivated for one of the pole parameters and slightly tuned for the other:

```
R> res_MH <- shrinkTVP(y ~ x1 + x2, data = data,
+    MH_tuning = list(a_xi_adaptive = FALSE,
+      a_tau_max_adapt = 0.001, a_tau_batch_size = 20),
+    display_progress = FALSE)
```

3.6. Posterior inference: Summarize the posterior distribution

The return value of `shrinkTVP()` is an object of type '`shrinkTVP`', which is a named list containing a variable number of elements, depending on the prior specification. For the default NG prior, the values are:

1. a list holding d `mcmc.tvp` objects (one for each $\boldsymbol{\beta}_j = (\beta_{j0}, \dots, \beta_{jT})$) containing the parameter draws in `beta`,
2. the parameter draws of $\boldsymbol{\beta} = (\beta_1, \dots, \beta_d)$ in `beta_mean`,

3. the parameter draws of $(\sqrt{\theta_1}, \dots, \sqrt{\theta_d})$ in `theta_sr`,
4. the parameter draws of $\tau_1^2, \dots, \tau_d^2$ in `tau2`,
5. the parameter draws of $\xi_1^2, \dots, \xi_d^2$, in `xi2`,
6. the parameter draws of a^ξ in `a_xi`,
7. the parameter draws of a^τ in `a_tau`,
8. the parameter draws for κ_B^2 in `kappa2_B`,
9. the parameter draws for λ_B^2 in `lambda2_B`,
10. the parameter draws of σ^2 in `sigma2`,
11. the parameter draws of C_0 in `C0`,
12. MH diagnostic values in `MH_diag`,
13. the prior hyperparameters in `priorvals`,
14. the design matrix, the response and the formula in `model`,
15. summary statistics for the parameter draws in `summaries` and objects required for the `LPDS()` function in `internals`.

When some parameters are fixed by the user, the corresponding output value is omitted. Additionally, increasing or decreasing the amount of levels in the hierarchy of the prior also changes which values are returned. For example, if `mod_type` is changed to `"triple"` and the learning of the tail parameters c^ξ and c^τ is not deactivated, then the output will also contain the respective parameter draws in `c_xi` and `c_tau`. In the SV case, the draws for the parameters of the SV model on the errors are contained in `sv_mu`, `sv_phi` and `sv_sigma`. For details, see Kastner (2016).

The two main tools for summarizing the output of `shrinkTVP()` are the `summary` and `plot` methods implemented for '`shrinkTVP`' objects. `summary` has two arguments beyond the '`shrinkTVP`' object itself, namely `digits` and `showprior`, which control the output displayed. `digits` indicates the number of decimal places to round the posterior summary statistics to, while `showprior` determines whether or not to show the prior distributions resulting from the user input. In the example below, the default `digits` value of 3 is used, while the prior specification is omitted. The output of `summary` consists of the mean, standard deviation, median, 95% highest posterior density region and effective sample size (ESS) for the non time-varying parameters.

```
R> summary(res, showprior = FALSE)
```

```
Summary of 5000 MCMC draws after burn-in of 5000.
Statistics of posterior draws of parameters (thinning = 1):

 param                    mean   sd     median HPD 2.5% HPD 97.5% ESS
 beta_mean_Intercept      0.171  0.464  0      -0.341   1.486     343
 beta_mean_x1            -0.227  0.162  -0.256 -0.481   0.016     148
 beta_mean_x2            -0.002  0.036  0      -0.095   0.07      3201

 abs(theta_sr_Intercept)  0.422  0.063  0.418  0.306    0.549     372
 abs(theta_sr_x1)        0.017  0.025    0.004  0        0.07     161
 abs(theta_sr_x2)        0.003  0.006    0      0        0.014    486

 tau2_Intercept          31.115 1250.202 0.001  0        6.082    4759
 tau2_x1                 6.854  122.203  0.075  0        4.747    4542
 tau2_x2                 2.517  110.132  0      0        0.092    5000

 xi2_Intercept           62.334 2807.843 0.241  0.009    9.647    5000
 xi2_x1                  0.137  2.318    0      0        0.095    3117
 xi2_x2                  0.018  0.579    0      0        0.003    5000

 a_xi                    0.085  0.037    0.079  0.023    0.158    511

 a_tau                   0.094  0.04     0.087  0.029    0.172    466

 kappa2_B                31.291 99.624   1.798  0        158.485  4233

 lambda2_B               71.779 230.802  3.023  0        371.266  1810

 sigma2                  0.992  0.124    0.983  0.753    1.232    1716

 C0                      1.708  0.622    1.634  0.584    2.892    5000
```

The `plot()` method can be used to visualize the posterior distribution estimated by `shrinkTVP()`. Aside from a `shrinkTVP` object, its main argument is `pars`, a character vector containing the names of the parameters to visualize. `plot()` will call either `plot.mcmc.tvp()` from the **shrinkTVP** package if the parameter is time-varying or `plot.mcmc()` from the **coda** package, if the parameter is non time-varying. The default value of `pars` is `c("beta")`, leading to `plot.mcmc.tvp()` being called on each of the β_{jt}, for $j = 1, \ldots, d$. See the code below for an example and Figure 1 for the corresponding output.

```
R> plot(res)
```

The `plot.mcmc.tvp()` method displays empirical posterior credible intervals of a time-varying parameter over time, i.e., β_{jt}, for $j = 1, \ldots, d$ and σ_t^2 in the case of stochastic volatility. By default, the pointwise 95% and 50% posterior credible intervals are displayed as shaded areas layered on top of one another, with the median represented by a black line, with an additional grey, dashed line at zero. To ensure that users have flexiblity in the plots created, a host of options are implemented for customisation. The bounds of the credible intervals can be modified through the `probs` input, allowing for different levels of uncertainty visualization. The arguments `quantlines` and `shaded` take boolean vectors as inputs, and determine if the corresponding credible intervals will be displayed through shading and/or lines. The shaded areas can be customised via the arguments `shadecol` and `shadealpha`, which determine the color and the degree of transparency of the shaded areas. The lines representing the quantiles can be adjusted through `quantlty`, `quantcol` and `quantlwd`, which modify the line type, color and line width, respectively. In the spirit of R, all of these arguments are vectorised and the supplied vectors are recycled in the typical R fashion if necessary. The first

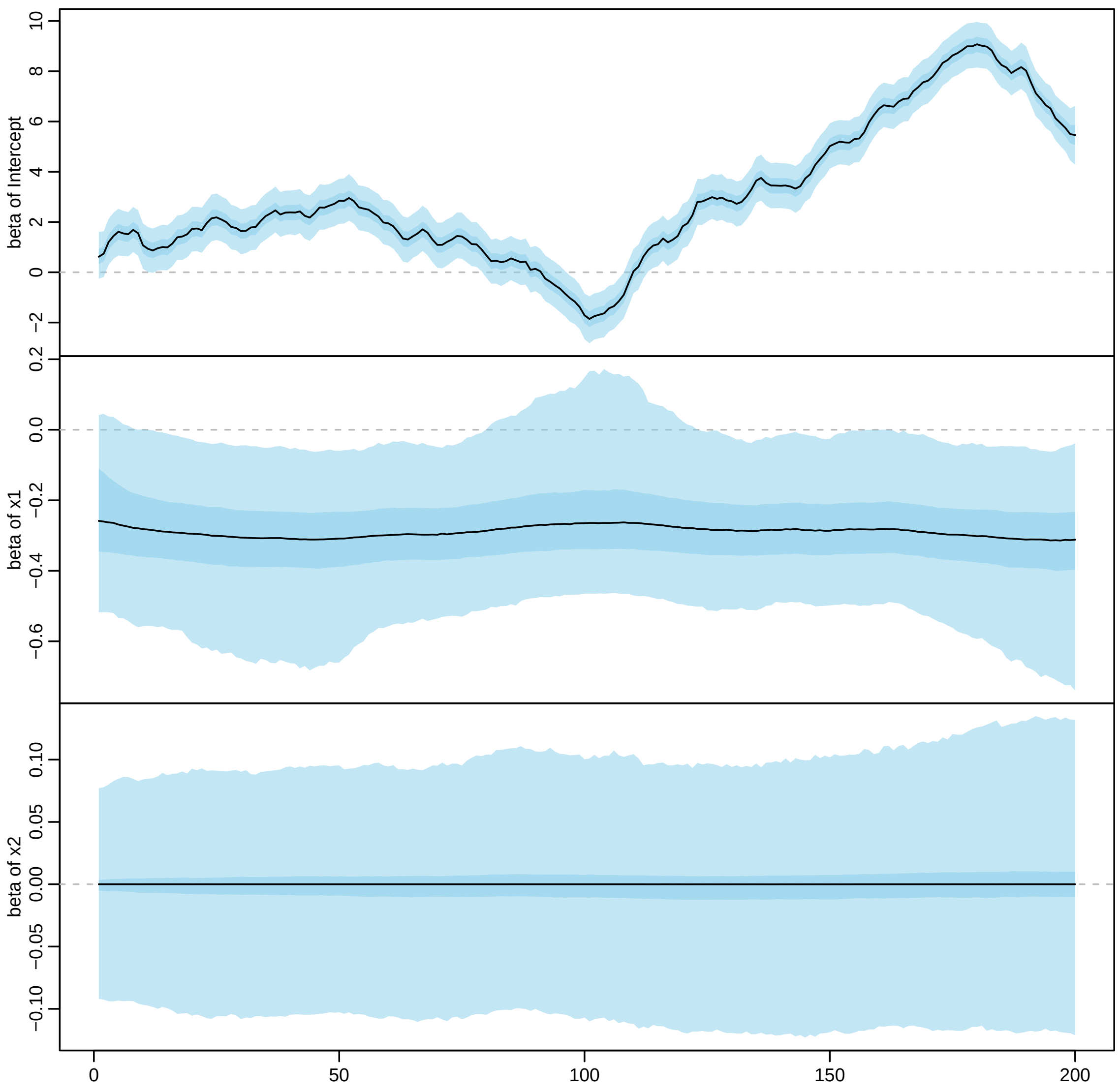

Figure 1: Visualization of the evolution of the time-varying parameter $\boldsymbol{\beta}_j = (\beta_{j0}, \dots, \beta_{jT}), j = 1, 2, 3$, over time $t = 0, \dots, T$, as provided by the `plot()` method. `plot()` is in turn calling `plot.mcmc.tvp()` on the individual '`mcmc.tvp`' objects. The median is displayed as a black line, and the shaded areas indicate the pointwise 95% and 50% posterior credible intervals.

element of these vectors is always applied to the outermost credible interval, the second to the second outermost and so forth. The horizontal line at zero can be similarly adjusted through `zerolty`, `zerolwd` and `zerocol` or entirely turned off by setting `drawzero` equal to `FALSE`. All further arguments are passed on to the standard `plot()` method, allowing for changes to the line representing the median and other plot modifications that users of R are familiar with. An example of possible customisation, with help from the package **RColorBrewer** (Neuwirth 2014), can be seen in the code below. The corresponding output is Figure 2.

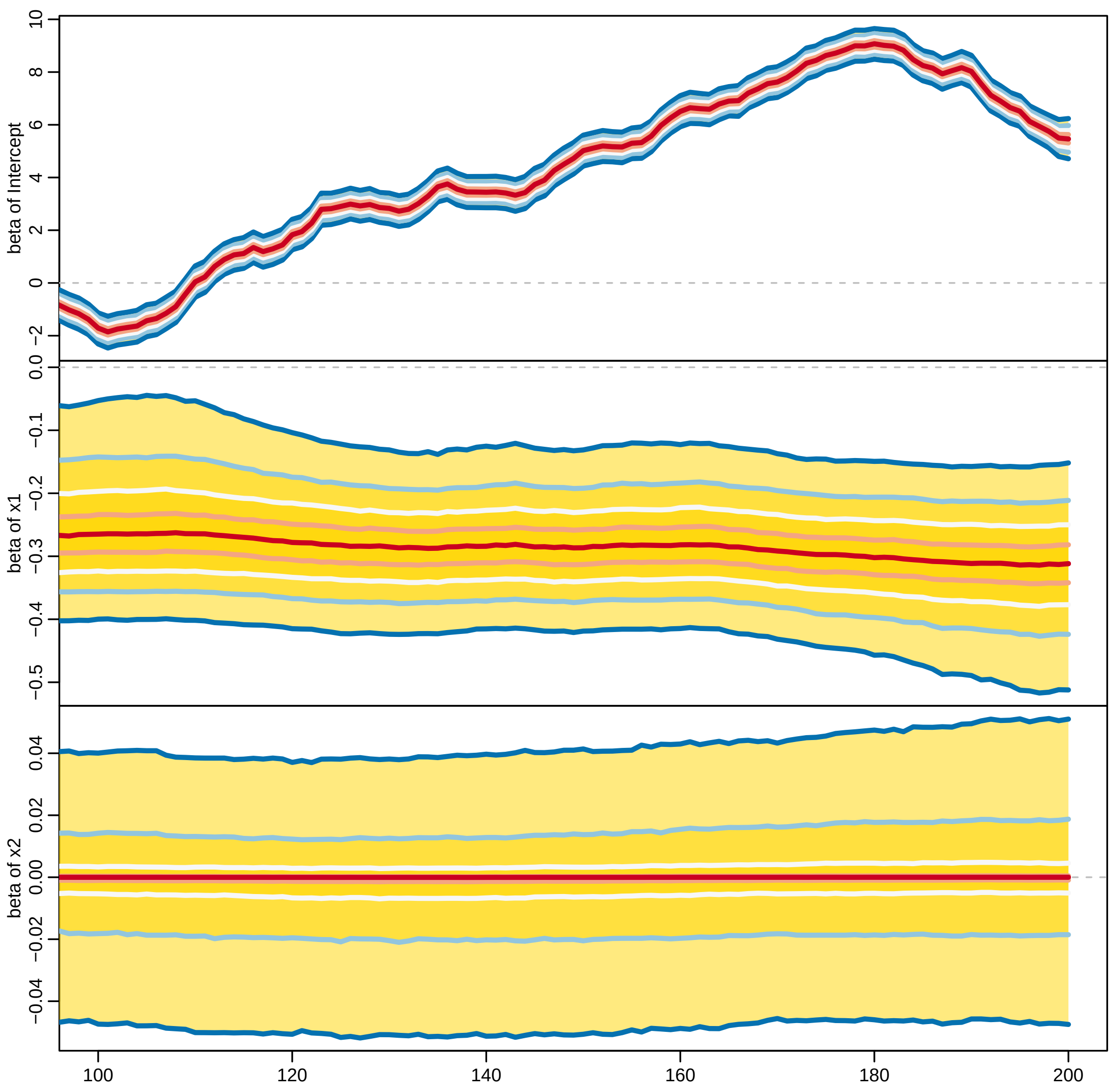

Figure 2: Visualization of the evolution of the time-varying parameter β_t over time $t = 100, \dots, 200$ for $j = 1, \dots, 3$. In this example, the x-axis of the plot was restricted with `xlim`, the color of the shaded areas was changed to yellow and colored solid lines have been added to delimit the credible intervals. The colored lines represent the median and the pointwise 10%, 20%, 30% 40%, 60%, 70%, 80%, and 90% quantiles.

```
R> library("RColorBrewer")
R> color <- brewer.pal(5, "RdBu")
R> plot(res, pars = "beta", xlim = c(100, 200),
+    probs = seq(0.1, 0.9, by = 0.1),
+    quantlines = TRUE, quantcol = color[5:2], quantlty = 1,
+    quantlwd = 3, col = color[1], lwd = 3, shadecol = "gold1")
```

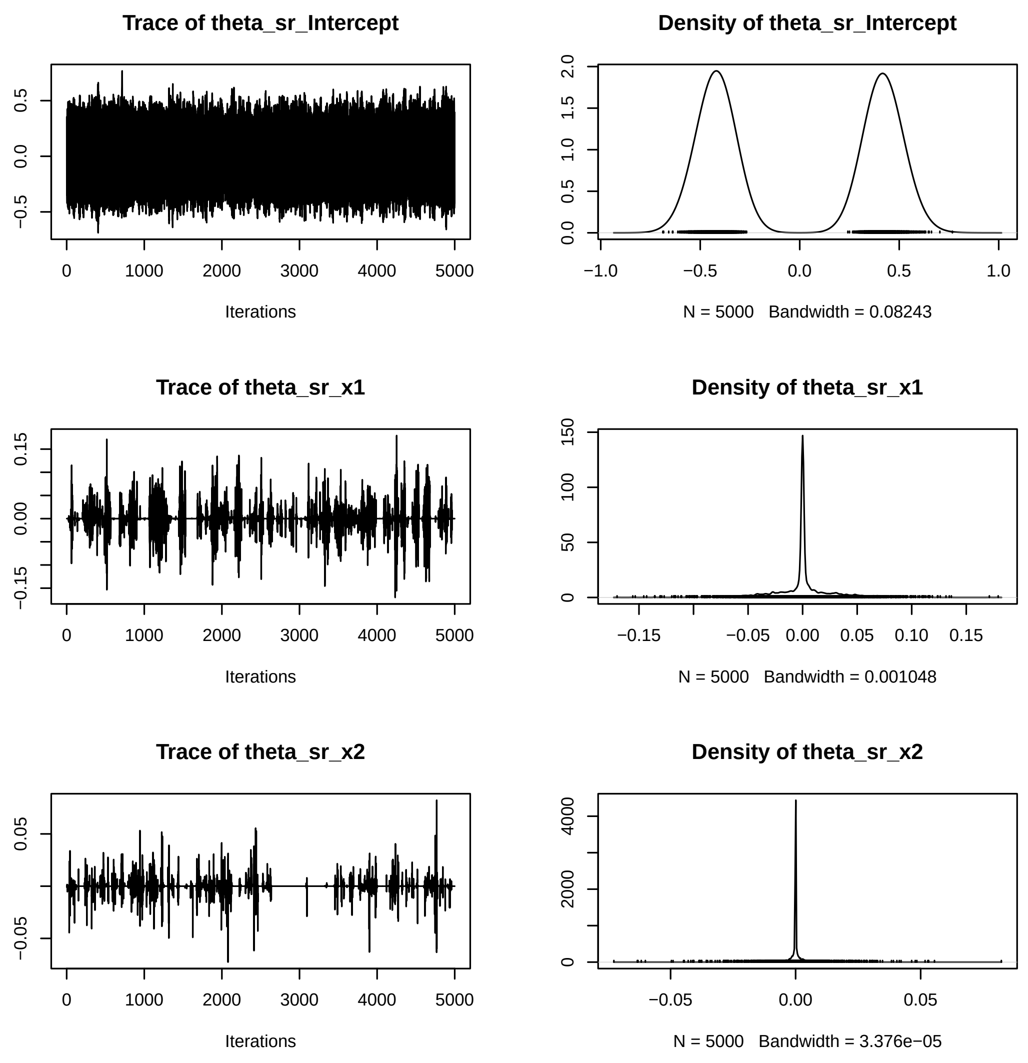

Figure 3: Trace plots (left column) and kernel density estimates of the posterior density (right column) for the parameters $\sqrt{\theta_1}, \dots, \sqrt{\theta_3}$, as provided by the `plot()` method. `plot()` is in turn calling **coda**'s `plot.mcmc()`.

To visualize other parameters via the `plot()` method, the user has to change the `pars` argument. `pars` can either be set to a single character object or to a vector of characters containing the names of the parameter draws to display. In the latter case, the `plot()` method will display groups of plots at a time, prompting the user to move on to the next series of plots, similarly to how **coda** handles long plot outputs. Naturally, as all parameter draws are converted to '**coda**' objects, any method from this package that users are familiar with (e.g., to check convergence) can be applied to the parameter draws contained in a '`shrinkTVP`' object. An example of this can be seen in Figure 3, where `pars = "theta_sr"`, changes the output to a graphical summary of the parameter draws of $\sqrt{\theta_1}, \dots, \sqrt{\theta_d}$, using **coda**'s `plot.mcmc()` function. To obtain Figure 3, one can run

```
R> plot(res, pars = "theta_sr")
```

4. Predictive performances and model comparison

Within a Bayesian framework, a natural way to predict a future observation is through its posterior predictive density. For this reason, log-predictive density scores (LPDSs) provide a means of assessing how well the model performs in terms of prediction on real data. The log-predictive density score for time t_0+1 is obtained by evaluating at y_{t_0+1} the log of the posterior predictive density obtained by fitting the model to the previous t_0 data points. Given the data up to time t_0, the posterior predictive density at time t_0+1 is given by

$$p(y_{t_0+1} \mid y_1, \dots, y_{t_0}, \boldsymbol{x}_{t_0+1}) = \int p(y_{t_0+1} \mid \boldsymbol{x}_{t_0+1}, \boldsymbol{\psi}) p(\boldsymbol{\psi} \mid y_1, \dots, y_{t_0}) d\boldsymbol{\psi}, \tag{17}$$

where $\boldsymbol{\psi}$ is the set of model parameters and latent variables up to t_0+1. For a TVP model with homoscedastic errors, $\boldsymbol{\psi} = (\tilde{\boldsymbol{\beta}}_0, \dots \tilde{\boldsymbol{\beta}}_{t_0+1}, \sqrt{\theta}_1, \dots, \sqrt{\theta_d}, \beta_1, \dots, \beta_d, \sigma^2)$, whereas for a TVP model with SV errors, $\boldsymbol{\psi} = (\tilde{\boldsymbol{\beta}}_0, \dots \tilde{\boldsymbol{\beta}}_{t_0+1}, \sqrt{\theta}_1, \dots, \sqrt{\theta_d}, \beta_1, \dots, \beta_d, \sigma^2_1, \dots, \sigma^2_{t_0+1})$. Given M samples from the posterior distribution of the parameters and latent variables, $p(\boldsymbol{\psi} \mid y_1, \dots, y_{t_0})$, Monte Carlo integration could be applied immediately to approximate (17). However, Bitto and Frühwirth-Schnatter (2019) propose a more efficient approximation of the predictive density, the so-called conditionally optimal Kalman mixture approximation which is obtained by analytically integrating out $\tilde{\boldsymbol{\beta}}_{t_0+1}$ from the likelihood at time t_0+1.

In the homoscedastic error case, given M samples from the posterior distribution of the parameters and the latent variables up to t_0, Monte Carlo integration of the resulting predictive density yields following mixture approximation,

$$\begin{aligned} p(y_{t_0+1} \mid y_1, \dots, y_{t_0}, \boldsymbol{x}_{t_0+1}) &\approx \frac{1}{M} \sum_{m=1}^{M} f_{\mathcal{N}}(y_{t_0+1}; \hat{y}^{(m)}_{t_0+1}, S^{(m)}_{t_0+1}), \\ \hat{y}^{(m)}_{t_0+1} &= \boldsymbol{x}_{t_0+1} \boldsymbol{\beta}^{(m)} + \boldsymbol{F}^{(m)}_{t_0+1} \boldsymbol{m}^{(m)}_{t_0}, \\ S^{(m)}_{t_0+1} &= \boldsymbol{F}^{(m)}_{t_0+1} (\boldsymbol{\Sigma}^{(m)}_{t_0} + I_d)(\boldsymbol{F}^{(m)}_{t_0+1})^\top + (\sigma^2)^{(m)}, \end{aligned} \tag{18}$$

where the conditional predictive densities are Gaussian and the conditional moments depend on the MCMC draws. The mean $\hat{y}^{(m)}_{t_0+1}$ and the variance $S^{(m)}_{t_0+1}$ are computed for the mth MCMC iteration from $\boldsymbol{F}_{t_0+1} = \boldsymbol{x}_{t_0+1} \text{Diag}(\sqrt{\theta_1}, \dots, \sqrt{\theta_d})$ and the mean $\boldsymbol{m}_{t_0}$ and the covariance matrix $\boldsymbol{\Sigma}_{t_0}$ of the posterior distribution of $\tilde{\boldsymbol{\beta}}_{t_0}$. These quantities can be obtained by iteratively calculating $\boldsymbol{\Sigma}_t$ and $\boldsymbol{m}_t$ up to time t_0, as described in McCausland *et al.* (2011):

$$\begin{aligned} &\boldsymbol{\Sigma}_1 = (\boldsymbol{\Omega}_{11})^{-1}, \qquad \boldsymbol{m}_1 = \boldsymbol{\Sigma}_1 \boldsymbol{c}_1, \\ &\boldsymbol{\Sigma}_t = (\boldsymbol{\Omega}_{tt} - \boldsymbol{\Omega}^\top_{t-1,t} \boldsymbol{\Sigma}_{t-1} \boldsymbol{\Omega}_{t-1,t})^{-1}, \qquad \boldsymbol{m}_t = \boldsymbol{\Sigma}_t (\boldsymbol{c}_t - \boldsymbol{\Omega}^\top_{t-1,t} \boldsymbol{m}_{t-1}). \end{aligned}$$

The quantities $\boldsymbol{c}_t$, $\boldsymbol{\Omega}_{tt}$ and $\boldsymbol{\Omega}_{t-1,t}$ for $t = 1, \dots, t_0$ are given in Appendix A.

For the SV case, it is still possible to analytically integrate out $\tilde{\boldsymbol{\beta}}_{t_0+1}$ from the likelihood at time t_0+1 conditional on a known value of $\sigma^2_{t_0+1}$, however it is not possible to integrate the likelihood with respect to both latent variables $\tilde{\boldsymbol{\beta}}_{t_0+1}$ and $\sigma^2_{t_0+1}$. Hence, at each MCMC iteration a draw is taken from the predictive distribution of $\sigma^2_{t_0+1} = \exp(h_{t_0+1})$, derived from Equation (3), and used to calculate the conditional predictive density of y_{t_0+1}. The approximation of the one-step ahead predictive density can then be obtained through the following steps:

1. for each MCMC draw of $(\mu, \phi, \sigma^2_\eta)^{(m)}$ and $h^{(m)}_{t_0}$, obtain a draw of $(\sigma^2_{t_0+1})^{(m)}$.

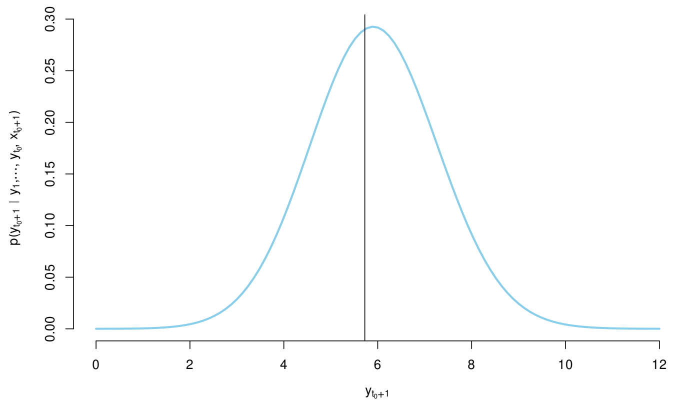

Figure 4: One-step ahead predictive density $p(y_{t_0+1} \mid y_1, \ldots, y_{t_0}, \boldsymbol{x}_{t_0+1})$ for a synthetic data set. The black vertical line represents the true realisation of y_{t_0+1}.

2. Calculate the conditionally optimal Kalman mixture approximation as in (18) with following slightly different values $S_{t_0+1}^{(m)}$:

$$S_{t_0+1}^{(m)} = \boldsymbol{F}_{t_0+1}^{(m)}(\boldsymbol{\Sigma}_{t_0}^{(m)} + I_d)(\boldsymbol{F}_{t_0+1}^{(m)})^\top + (\sigma_{t_0+1}^2)^{(m)},$$

   where $\boldsymbol{F}_{t_0+1}$ and $\boldsymbol{\Sigma}_{t_0}$ are the same as defined above.

These calculations can be performed by the `LPDS()` function, based on a fitted TVP model resulting from a call to `shrinkTVP()`. The function's arguments are an object of class '`shrinkTVP`' and '`data_test`', a data frame with one row, containing covariates and response at time $t_0 + 1$. The following snippet of code fits a '`shrinkTVP`' model to synthetic data up to $T - 1$, and then calculates the LPDS at time T. The obtained LPDS score is then displayed. For an example on how to calculate LPDSs for k points in time, please see Section 5.

```
R> res_LPDS <- shrinkTVP(y ~ x1 + x2, data = data[1:(nrow(data) - 1),],
+    display_progress = FALSE)
R> LPDS(res_LPDS, data[nrow(data), ])
```

```
[1] -1.237525
```

An additional functionality provided by the package **shrinkTVP** is the evaluation of the one-step ahead predictive density through the function `eval_pred_dens()`.

It takes as inputs an object of class '`shrinkTVP`', a one row data frame containing $\boldsymbol{x}_{t_0+1}$ and a point, or vector of points, at which the predictive density is to be evaluated. It returns a vector of the same length, containing the value of the density at the points the user supplied. An example of this can be seen in the code below.

```
R> eval_pred_dens(1:3, res_LPDS, data[nrow(data), ])
```

```
[1] 0.0004201221 0.0044530770 0.0286555720
```

Thanks to its vectorised nature, `eval_pred_dens()` can be plugged directly into functions that expect an expression that evaluates to the length of the input, such as the `curve()` function from the **graphics** (R Core Team 2021) package. The following snippet of code exploits this behaviour to plot the posterior predictive density. The result can be seen in Figure 4.

```
R> curve(eval_pred_dens(x, res_LPDS, data[nrow(data), ]), to = 12,
+    ylab = bquote("p(" * y[t[0]+1] * "\uff5c" * y[1] * ","
+    * ldots * "," ~ y[t[0]] * "," ~ x[t[0]+1] * ")"),
+    xlab = expression(y[t[0]+1]), lwd = 2.5, col = "skyblue", axes = FALSE)
R> abline(v = data$y[nrow(data)])
R> axis(1)
R> axis(2)
```

5. Predictive exercise: `usmacro` dataset

In the following, we provide a brief demonstration on how to use the **shrinkTVP** package on real data and compare different prior specifications via LPDSs. Specifically, we consider the `usmacro.update` dataset from the **bvarsv** package (Krüger 2015). The dataset `usmacro.update` contains the inflation rate, unemployment rate and treasury bill interest rate for the United States, from 1953:Q1 to 2015:Q2, that is $T = 250$. The same dataset up to 2001:Q3 was used by Primiceri (2005). The response variable is the inflation rate `inf`, while the predictors are the lagged inflation rate `inf_lag`, the lagged unemployed rate `une_lag` and the lagged treasury bill interest `tbi_lag`. We construct our dataset as follows:

```
R> library("bvarsv")
R> data("usmacro.update", package = "bvarsv")
R> lags <- usmacro.update[1:(nrow(usmacro.update) - 1), ]
R> colnames(lags) <- paste0(colnames(lags), "_lag")
R> us_data <- data.frame(inf = usmacro.update[2:nrow(usmacro.update), "inf"],
+    lags)
```

In the snippet of code below, we estimate a TVP model with a fully hierarchical NG prior for 60000 iterations, with a thinning of 10 and a burn-in of 10000, hence keeping 5000 posterior draws.

```
R> us_res <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, us_data,
+    niter = 60000, nburn = 10000, nthin = 10,
+    display_progress = FALSE)
```

Once we have fit the model, we can perform posterior inference by using the `summary()` and `plot()` methods. The summary is shown below, while Figure 5 shows the paths of $\boldsymbol{\beta}_t$ evolving over time, and Figure 6 displays the trace plots (left column) and posterior densities (right column) of $\sqrt{\theta_1}, \ldots, \sqrt{\theta_4}$ obtained via the `plot` method.

```
R> summary(us_res, showprior = FALSE)
```

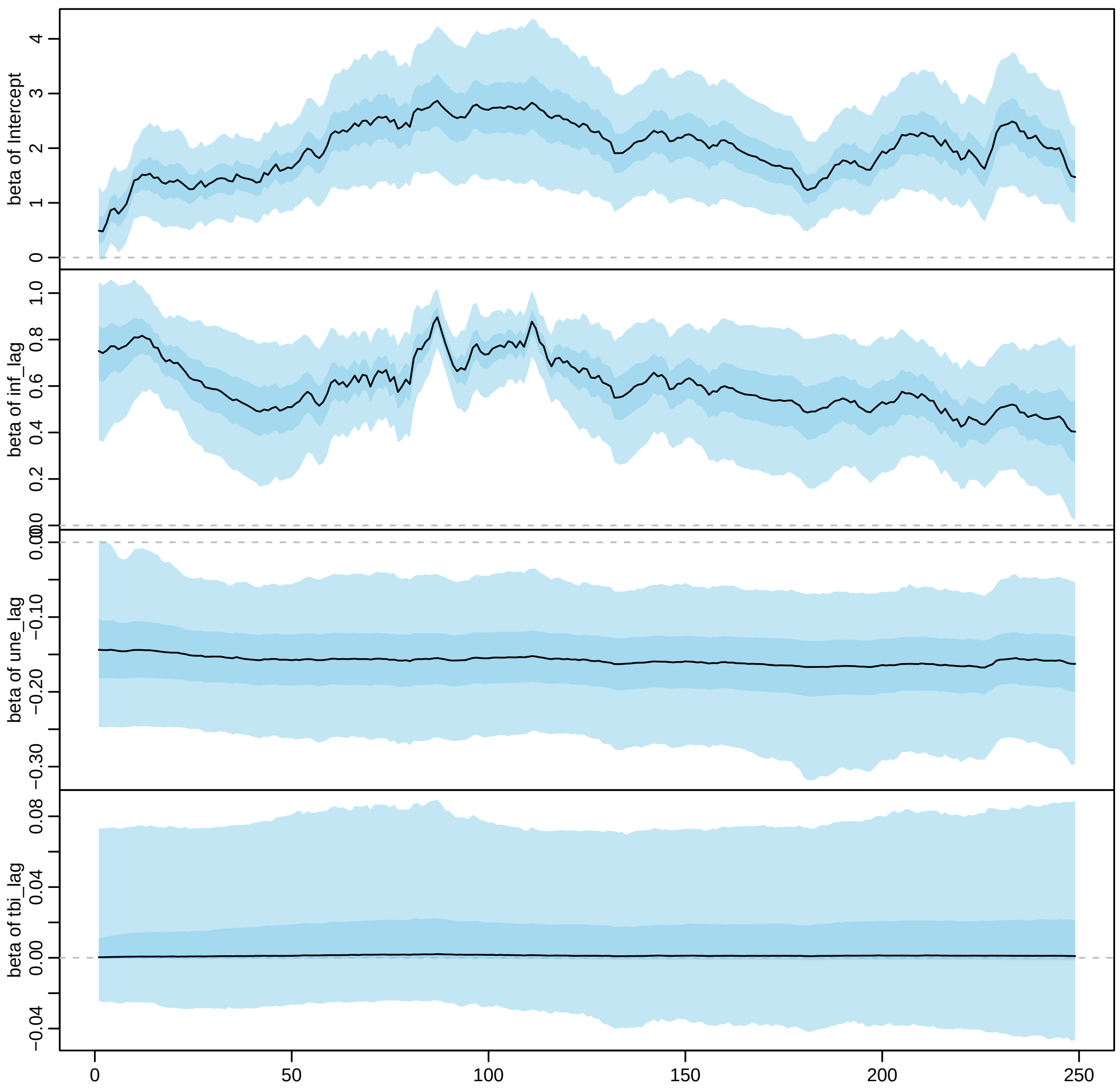

Figure 5: Visualization of the evolution of the time-varying parameter $\boldsymbol{\beta}_j = (\beta_{j0}, \dots, \beta_{jT})$ over time $t = 0, \dots, T$ for $j = 1, \dots, 4$ for the `usmacro.update` dataset. The median is displayed as a black line, and the shaded areas indicate the pointwise 95% and 50% posterior credible intervals.

```
Summary of 50000 MCMC draws after burn-in of 10000.
Statistics of posterior draws of parameters (thinning = 10):

 param                    mean     sd       median HPD 2.5% HPD 97.5% ESS
 beta_mean_Intercept      0.404    0.433    0.295  -0.076   1.304     481
 beta_mean_inf_lag        0.73     0.188    0.742  0.347    1.077     696
 beta_mean_une_lag        -0.136   0.066    -0.143 -0.237   0.006     268
 beta_mean_tbi_lag        0.008    0.023    0      -0.027   0.067     700
```

```
abs(theta_sr_Intercept) 0.143   0.025    0.143  0.093    0.19      1128
abs(theta_sr_inf_lag)   0.043   0.006    0.043  0.031    0.056     2280
abs(theta_sr_une_lag)   0.004   0.005    0.001  0        0.016     101
abs(theta_sr_tbi_lag)   0.001   0.003    0      0        0.007     451

tau2_Intercept          13.984  278.347  0.152  0        12.146    5000
tau2_inf_lag            29.851  853.271  0.735  0        23.376    5000
tau2_une_lag            3.881   100.588  0.041  0        2.64      5000
tau2_tbi_lag            0.107   2.297    0      0        0.086     4178

xi2_Intercept           26.151  1759.323 0.032  0.001    0.936     5000
xi2_inf_lag             11.818  811.958  0.004  0        0.203     5000
xi2_une_lag             0.029   1        0      0        0.006     3942
xi2_tbi_lag             0.012   0.497    0      0        0.001     5000

a_xi                    0.096   0.041    0.09   0.024    0.176     748

a_tau                   0.105   0.042    0.098  0.031    0.188     1510

kappa2_B                117.598 269.506  21.01  0        552.254   4683

lambda2_B               8.251   25.944   1.126  0        37.14     4801

sigma2                  0.018   0.006    0.018  0.008    0.029     1467

C0                      0.127   0.062    0.117  0.029    0.252     2436
```

It appears clear by looking at Figure 5 that the intercept and the parameter associated with the lagged inflation rate are time-varying, while the parameters associated with the lagged treasury bill interest rate and the lagged unemployment rate are relatively constant. This can be confirmed by looking at the posterior distributions of the corresponding standard deviations, displayed in Figure 6. The posterior densities of the standard deviations associated with the intercept and the lagged inflation are bimodal, with very little mass around zero. This bimodality results from the non-identifiability of the sign of the standard deviation. As a convenient side effect, noticeable bimodality in the density plots of the posterior distribution $p(\sqrt{\theta}_j \mid \mathbf{y})$ of the standard deviations $\sqrt{\theta}_j$ is a strong indication of time variability in the associated parameter β_{jt}. Conversely, the posterior densities of the standard deviations associated with the lagged unemployment and the lagged treasury bill interest rate have a pronounced spike at zero, indicating strong model evidence in favor of constant parameters. Moreover, the path of the parameter of the treasury bill interest rate is centered at zero, indicating that this parameter is neither time-varying nor significant.

In order to compare the predictive performances of different shrinkage priors, we calculate one-step ahead LPDSs for the last 50 points in time for eleven different prior choices: (1) the full hierarchical NGG prior, (2) the hierarchical NGG prior with fixed $a^\xi = a^\tau = c^\xi = c^\tau = 0.1$, (3) the NGG prior with $a^\xi = a^\tau = c^\xi = c^\tau = 0.1$ and $\kappa_B^2 = \lambda_B^2 = 20$, (4) the hierarchical horseshoe prior, (5) the horseshoe prior $\kappa_B^2 = \lambda_B^2 = 20$, (6) the full hierarchical NG prior,

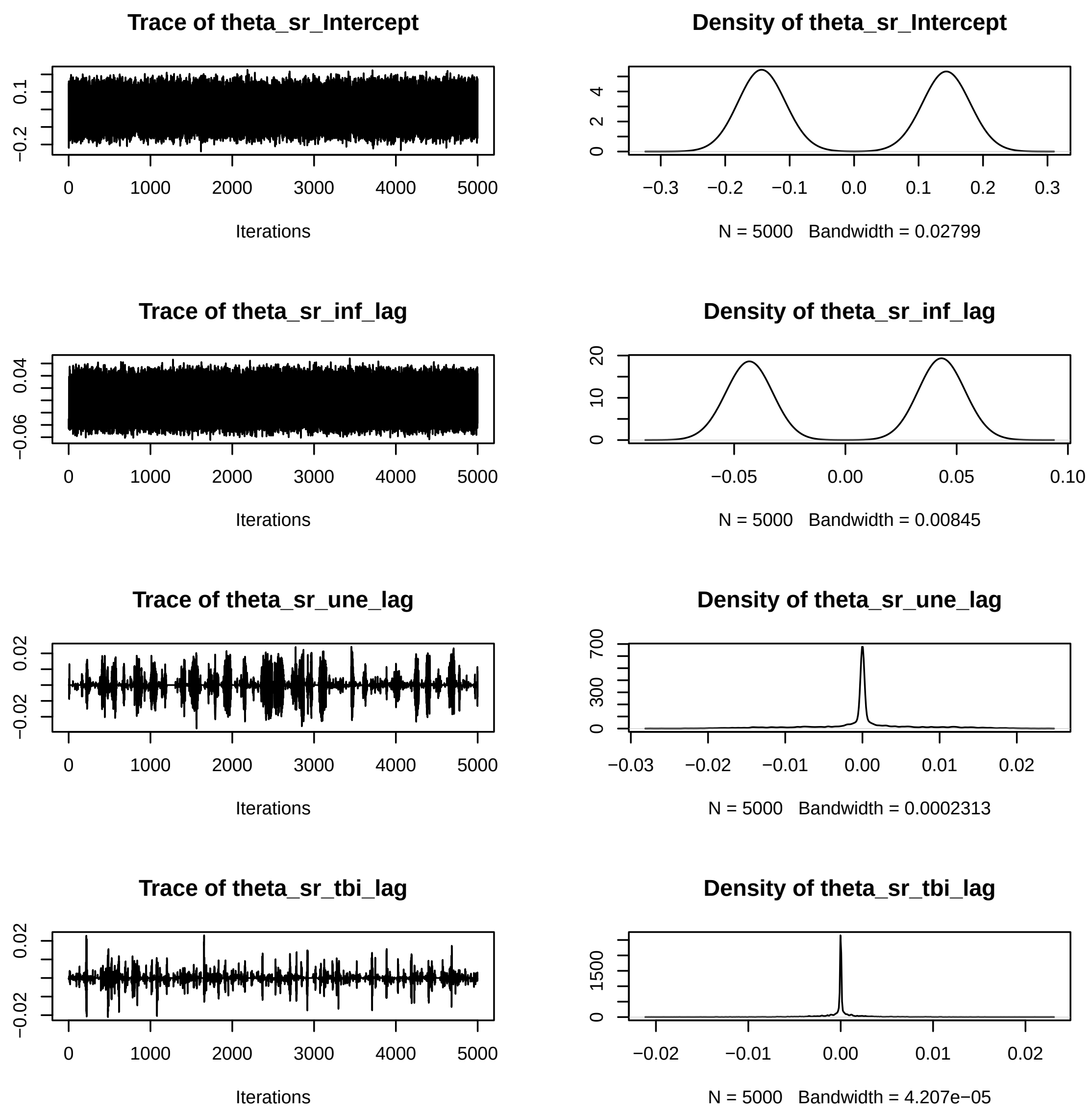

Figure 6: Trace plots (left column) and kernel density estimates of the posterior density (right column) for the parameters $\sqrt{\theta_1}, \dots, \sqrt{\theta_4}$ for the `usmacro.update` dataset.

(7) the hierarchical NG prior with fixed $a^\xi = a^\tau = 0.1$, (8) the NG prior with $a^\xi = a^\tau = 0.1$ and $\kappa_B^2 = \lambda_B^2 = 20$, (9) the hierarchical Bayesian Lasso, and (10) the Bayesian Lasso with $\kappa_B^2 = \lambda_B^2 = 20$ and (11) ridge regression with $\kappa_B^2 = \lambda_B^2 = 20$. Figure 7 shows the cumulative LPDSs for the last 50 quarters of the `usmacro.update` dataset. The default prior, the fully hierarchical NG prior on both the β_j's and the $\sqrt{\theta_j}$'s, is among the best performing priors in terms of prediction. In Appendix B we show how to obtain LPDSs for different models and points in time, using the packages **foreach** (Microsoft and Weston 2020b; Kane, Emerson, and Weston 2013) and **doParallel** (Microsoft and Weston 2020a).

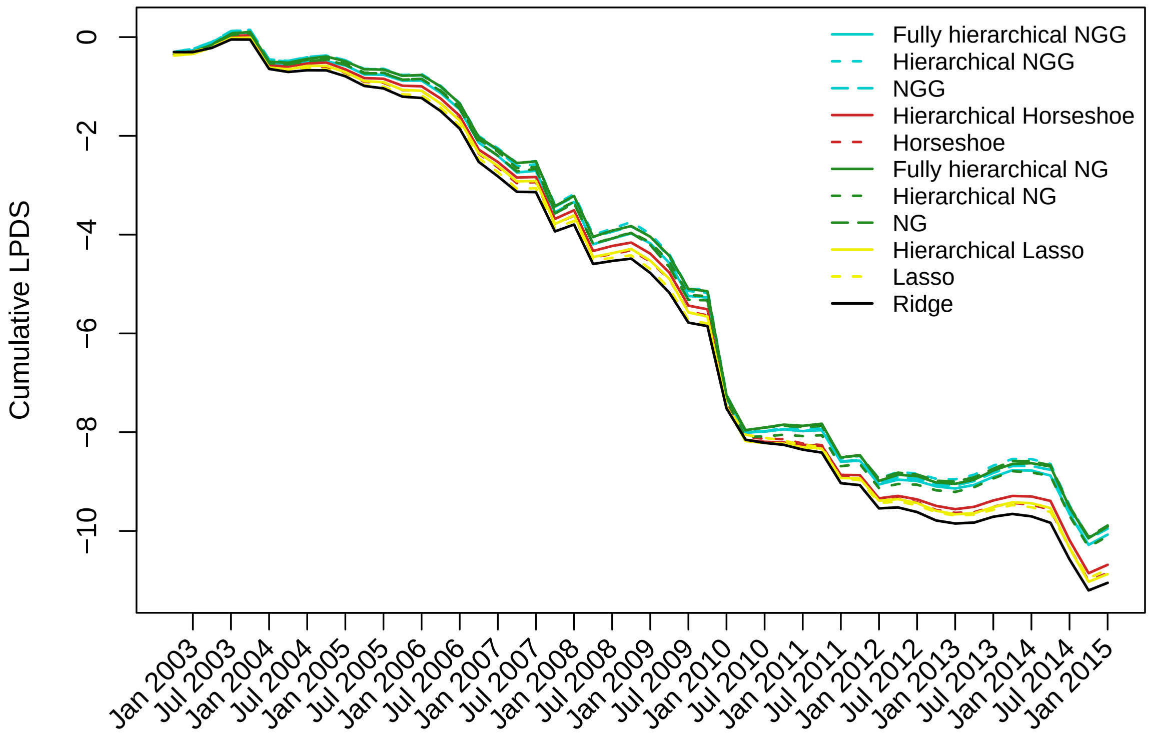

Figure 7: Cumulative LPDSs for the last 50 quarters of the `usmacro.update` dataset, for eleven different shrinkage priors: (1) the full hierarchical NGG prior, (2) the hierarchical NGG prior with fixed $a^\xi = a^\tau = c^\xi = c^\tau = 0.1$, (3) the NGG prior with $a^\xi = a^\tau = c^\xi = c^\tau = 0.1$ and $\kappa_B^2 = \lambda_B^2 = 20$, (4) the hierarchical horseshoe prior, (5) the horseshoe prior $\kappa_B^2 = \lambda_B^2 = 20$, (6) the full hierarchical NG prior, (7) the hierarchical NG prior with fixed $a^\xi = a^\tau = 0.1$, (8) the NG prior with $a^\xi = a^\tau = 0.1$ and $\kappa_B^2 = \lambda_B^2 = 20$, (9) the hierarchical Bayesian Lasso, and (10) the Bayesian Lasso with $\kappa_B^2 = \lambda_B^2 = 20$ and (11) ridge regression with $\kappa_B^2 = \lambda_B^2 = 20$.

6. Conclusions

The goal of this paper was to introduce the reader to the functionality of the R package **shrinkTVP** (Knaus *et al.* 2021). This R package provides a fully Bayesian approach for statistical inference in TVP models with shrinkage priors. On the one hand, the package provides an easy entry point for users who want to pass on only their data in a first step of exploring TVP models for their specific application context. Running the function `shrinkTVP()` under the default model with a fully hierarchical NG shrinkage prior with predefined hyperparameters, estimation of a TVP model becomes as easy as using the well-known function `lm()` for a standard linear regression model. On the other hand, exploiting numerous advanced options of the package, the more experienced user can also explore alternative model specifications such as the Bayesian Lasso or the horseshoe prior and use log-predictive density scores to compare various model specifications.

Various examples of the usage of **shrinkTVP** were given, and the `summary` and `plot` methods for straightforward posterior inference were illustrated. Furthermore, a predictive exercise with the dataset `usmacro.updade` from the package **bvarsv** was performed, with a focus on the calculation of LPDSs using `shrinkTVP()`. The default model in **shrinkTVP** showed better performance than its competitors in terms of cumulative LPDSs. While these examples were confined to univariate responses, the package can also be applied in a multivariate context, for instance to the sparse TVP Cholesky SV model considered in Bitto and Frühwirth-Schnatter (2019), exploiting a representation of this model as a system of independent TVP models with univariate responses.

Computational details

The results in this paper were obtained with R 4.1.2, **g++** compiler version 7.5.0 for C++ and **openBLAS** version 0.3.8, running on an Ubuntu 20.04.3 LTS computer with an Intel Core i5-8350U CPU. The R packages used that are relevant to random number generation were **GIGrvg** version 0.5, **Rcpp** version 1.0.7, **RcppArmadillo** version 0.10.6.0.0 and **stochvol** version 3.1.0.

Replication code including random seeds is provided as part of the supplementary materials. Note that, due to the stochastic nature of inference based on MCMC simulations, exact replication of the results is only possible if all the aforementioned variables are held constant. Under different conditions, slightly different numbers might be obtained. These are equivalent to our results according to the law of large numbers and lead to the same qualitative conclusions.

References

Bai R, Ghosh M (2018). ***MBSP**: Multivariate Bayesian Model with Shrinkage Priors.* R package version 1.0, URL `https://CRAN.R-project.org/package=MBSP`.

Bai R, Ghosh M (2019). ***NormalBetaPrime**: Normal Beta Prime Prior.* R package version 2.2, URL `https://CRAN.R-project.org/package=NormalBetaPrime`.

Belmonte MAG, Koop G, Korobolis D (2014). "Hierarchical Shrinkage in Time-Varying Parameter Models." *Journal of Forecasting*, **33**, 80–94. `doi:10.1002/for.2276`.

Bhadra A, Datta J, Polson NG, Willard B (2019). "Lasso Meets Horseshoe: A Survey." *Statistical Science*, **34**, 405–427. `doi:10.1214/19-sts700`.

Bitto A, Frühwirth-Schnatter S (2019). "Achieving Shrinkage in a Time-Varying Parameter Model Framework." *Journal of Econometrics*, **210**, 75–97. `doi:10.1016/j.jeconom.2018.11.006`.

Breheny P, Huang J (2011). "Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection." *The Annals of Applied Statistics*, **5**(1), 232–253. `doi:10.1214/10-aoas388`.

Cadonna A, Frühwirth-Schnatter S, Knaus P (2020). "Triple the Gamma – A Unifying Shrinkage Prior for Variance and Variable Selection in Sparse State Space and TVP Models." *Econometrics*, **8**(2), 20. `doi:10.3390/econometrics8020020`.

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1). doi:10.18637/jss.v076.i01.

Carvalho CM, Polson NG, Scott JG (2009). "Handling Sparsity via the Horseshoe." In D Van Dyk, M Welling (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 73–80. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. URL https://proceedings.mlr.press/v5/carvalho09a.html.

Casas I, Fernandez-Casal R (2021). ***tvReg**: Time-Varying Coefficients Linear Regression For Single and Multi-Equations.* R package version 0.5.6, URL https://CRAN.R-project.org/package=tvReg.

Dangl T, Halling M (2012). "Predictive Regressions with Time-Varying Coefficients." *Journal of Financial Economics*, **106**, 157–181. doi:10.1016/j.jfineco.2012.04.003.

Del Negro M, Primiceri G (2015). "Time Varying Structural Vector Autoregressions and Monetary Policy: A Corrigendum." *Review of Economic Studies*, **82**(4), 1342–1345. doi:10.1093/restud/rdv024.

Dunkler D, Sauerbrei W, Heinze G (2016). "Global, Parameterwise and Joint Shrinkage Factor Estimation." *Journal of Statistical Software*, **69**(8), 1–19. doi:10.18637/jss.v069.i08.

Eddelbuettel D, Balamuta JJ (2018). "Extending R with C++: A Brief Introduction to **Rcpp**." *The American Statistician*, **72**(1), 28–36. doi:10.1080/00031305.2017.1375990.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models Via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Frühwirth-Schnatter S, Wagner H (2010). "Stochastic Model Specification Search for Gaussian and Partially Non-Gaussian State Space Models." *Journal of Econometrics*, **154**, 85–100. doi:10.1016/j.jeconom.2009.07.003.

Gramacy RB (2019). ***monomvn**: Estimation for Multivariate Normal and Student t Data with Monotone Missingness.* R package version 1.9-13, URL https://CRAN.R-project.org/package=monomvn.

Griffin JE, Brown PJ (2010). "Inference with Normal-Gamma Prior Distributions in Regression Problems." *Bayesian Analysis*, **5**, 171–188. doi:10.1214/10-ba507.

Griffin JE, Brown PJ (2017). "Hierarchical Shrinkage Priors for Regression Models." *Bayesian Analysis*, **12**, 135–159. doi:10.1214/15-ba990.

Helske J (2021). ***walker***: *Bayesian Generalized Linear Models with Time-Varying Coefficients.* R package version 1.0.3, URL `https://CRAN.R-project.org/package=walker`.

Hosszejni D, Kastner G (2021). "Modeling Univariate and Multivariate Stochastic Volatility in R with **stochvol** and **factorstochvol**." *Journal of Statistical Software*, **100**(12), 1–34. `doi:10.18637/jss.v100.i12`.

Jacquier E, Polson NG, Rossi PE (1994). "Bayesian Analysis of Stochastic Volatility Models." *Journal of Business & Economic Statistics*, **12**, 371–417. `doi:10.1198/073500102753410408`.

Kane MJ, Emerson J, Weston S (2013). "Scalable Strategies for Computing with Massive Data." *Journal of Statistical Software*, **55**(14), 1–19. `doi:10.18637/jss.v055.i14`.

Kastner G (2016). "Dealing with Stochastic Volatility in Time Series Using the R Package **stochvol**." *Journal of Statistical Software*, **69**, 1–30. `doi:10.18637/jss.v069.i05`.

Kastner G, Frühwirth-Schnatter S (2014). "Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Estimation of Stochastic Volatility Models." *Computational Statistics & Data Analysis*, **76**, 408–423. `doi:10.1016/j.csda.2013.01.002`.

Kastner G, Frühwirth-Schnatter S, Lopes HF (2017). "Efficient Bayesian Inference for Multivariate Factor Stochastic Volatility Models." *Journal of Computational and Graphical Statistics*, **26**, 905–917. `doi:10.1080/10618600.2017.1322091`.

Knaus P, Bitto-Nemling A, Cadonna A, Frühwirth-Schnatter S (2021). ***shrinkTVP***: *Efficient Bayesian Inference for Time-Varying Parameter Models with Shrinkage.* R package version 2.0.4, URL `https://CRAN.R-project.org/package=shrinkTVP`.

Krüger F (2015). ***bvarsv***: *Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters.* R package version 1.1, URL `https://CRAN.R-project.org/package=bvarsv`.

Leydold J, Hörmann W (2017). ***GIGrvg***: *Random Variate Generator for the GIG Distribution.* R package version 0.5, URL `https://CRAN.R-project.org/package=GIGrvg`.

McCausland WJ, Miller S, Pelletier D (2011). "Simulation Smoothing for State Space Models: A Computational Efficiency Analysis." *Computational Statistics & Data Analysis*, **55**, 199–212. `doi:10.1016/j.csda.2010.07.009`.

Microsoft, Weston S (2020a). ***doParallel***: *Foreach Parallel Adaptor for the* ***parallel*** *Package.* R package version 1.0.16, URL `https://CRAN.R-project.org/package=doParallel`.

Microsoft, Weston S (2020b). ***foreach***: *Provides Foreach Looping Construct for R.* R package version 1.5.1, URL `https://CRAN.R-project.org/package=foreach`.

Nakajima J (2011). "Time-Varying Parameter VAR Model with Stochastic Volatility: An Overview of Methodology and Empirical Applications." *Monetary and Economic Studies*, **29**, 107–142.

Nakano J, Nakama E (2021). ***RhpcBLASctl***: *Control the Number of Threads on BLAS.* R package version 0.21-247.1, URL `https://CRAN.R-project.org/package=RhpcBLASctl`.

Neuwirth E (2014). ***RColorBrewer**: ColorBrewer Palettes.* R package version 1.1-2, URL https://CRAN.R-project.org/package=RColorBrewer.

Park T, Casella G (2008). "The Bayesian Lasso." *Journal of the American Statistical Association*, **103**, 681–686. doi:10.1198/016214508000000337.

Petris G (2010). "An R Package for Dynamic Linear Models." *Journal of Statistical Software*, **36**(12), 1–16. doi:10.18637/jss.v036.i12.

Petris G, Petrone S, Campagnoli P (2009). *Dynamic Linear Models with R*. Springer-Verlag, New York. doi:10.1007/b135794.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://www.R-project.org/doc/Rnews/.

Plummer M, Best N, Cowles K, Vines K, Sarkar D, Bates D, Almond R, Magnusson A (2020). ***coda**: Output Analysis and Diagnostics for MCMC.* R package version 0.19-4, URL https://CRAN.R-project.org/package=coda.

Primiceri G (2005). "Time Varying Structural Vector Autoregressions and Monetary Policy." *Review of Economic Studies*, **72**, 821–852. doi:10.1111/j.1467-937x.2005.00353.x.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Roberts GO, Rosenthal JS (2009). "Examples of Adaptive MCMC." *Journal of Computational and Graphical Statistics*, **18**, 349–367. doi:10.1198/jcgs.2009.06134.

Ryan JA, Ulrich JM (2020). ***xts**: eXtensible Time Series.* R package version 0.12.1, URL https://CRAN.R-project.org/package=xts.

Scott SL (2021). ***bsts**: Bayesian Structural Time Series.* R package version 0.9.7, URL https://CRAN.R-project.org/package=bsts.

Simpson M, Niemi J, Roy V (2017). "Interweaving Markov Chain Monte Carlo Strategies for Efficient Estimation of Dynamic Linear Models." *Journal of Computational and Graphical Statistics*, **26**(1), 152–159. doi:10.1080/10618600.2015.1105748.

Sims CA (2001). "Evolving Post-World War II U.S. Inflation Dynamics: Comment." *NBER Macroeconomics Annual*, **16**, 373–379. ISSN 08893365, 15372642. doi:10.1086/654452.

Van der Pas S, Scott J, Chakraborty A, Bhattacharya A (2019). ***horseshoe**: Implementation of the Horseshoe Prior.* R package version 0.2.0, URL https://CRAN.R-project.org/package=horseshoe.

Vihola M, Helske J, Franks J (2017). "Importance Sampling Type Estimators Based on Approximate Marginal MCMC." arXiv:1609.02541 [stat.CO], URL https://arxiv.org/abs/1609.02541.

Yu Y, Meng XL (2011). "To Center or Not to Center: That Is Not the Question - An Ancillarity-Suffiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency." *Journal of Computational and Graphical Statistics*, **20**, 531–615. doi:10.1198/jcgs.2011.203main.

Zeileis A, Grothendieck G (2005). "**zoo**: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software*, **14**(6), 1–27. doi:10.18637/jss.v014.i06.

Zeileis A, Grothendieck G, Ryan JA (2021). ***zoo**: S3 Infrastructure for Regular and Irregular Time Series (Z's Ordered Observations)*. R package version 1.8-9, URL https://CRAN.R-project.org/package=zoo.

Zeng Y, Breheny P (2017). "The **biglasso** Package: A Memory- And Computation-Efficient Solver for Lasso Model Fitting with Big Data in R." arXiv:1701.05936 [stat.CO], URL https://arxiv.org/abs/1701.05936.

A. Full conditional distribution of the latent states

Let $y_t^\star = y_t - \boldsymbol{x}_t\boldsymbol{\beta}$ and $\mathbf{F}_t = \boldsymbol{x}_t\text{Diag}\left(\sqrt{\theta_1}, \ldots, \sqrt{\theta_d}\right)$ for $t = 1, \ldots, T$. Conditional on all other variables, the joint density for the state process $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_0, \tilde{\boldsymbol{\beta}}_1, \ldots, \tilde{\boldsymbol{\beta}}_T)$ is multivariate normal. This distribution can be written in terms of the tri-diagonal precision matrix $\boldsymbol{\Omega}$ and the mean vector $\mathbf{c}$ (McCausland *et al.* 2011):

$$\tilde{\boldsymbol{\beta}} \mid \boldsymbol{\beta}, \mathbf{Q}, \sigma_1^2, \ldots, \sigma_T^2, y_1^\star, \ldots y_T^\star \sim \mathcal{N}_{(T+1)d}\left(\boldsymbol{\Omega}^{-1}\mathbf{c}, \boldsymbol{\Omega}^{-1}\right) \tag{19}$$

where:

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Omega}_{00} & \boldsymbol{\Omega}_{01} & 0 & & & \\ \boldsymbol{\Omega}_{01}^\top & \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} & 0 & 0 & \\ 0 & \boldsymbol{\Omega}_{12}^\top & \boldsymbol{\Omega}_{22} & \boldsymbol{\Omega}_{23} & \ddots & \vdots \\ & 0 & \boldsymbol{\Omega}_{23}^\top & \ddots & \ddots & 0 \\ & \vdots & \ddots & \ddots & \boldsymbol{\Omega}_{T-1,T-1} & \boldsymbol{\Omega}_{T-1,T} \\ & 0 & \ldots & 0 & \boldsymbol{\Omega}_{T-1,T}^\top & \boldsymbol{\Omega}_{TT} \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_T \end{bmatrix}.$$

In this representation, each submatrix $\boldsymbol{\Omega}_{ts}$ is a matrix of dimension $d \times d$ defined as

$$\begin{aligned} \boldsymbol{\Omega}_{00} &= 2I_d, \\ \boldsymbol{\Omega}_{tt} &= \mathbf{F}_t^\top \mathbf{F}_t/\sigma_t^2 + 2I_d, \quad t = 1, \ldots, T-1, \\ \boldsymbol{\Omega}_{TT} &= \mathbf{F}_T^\top \mathbf{F}_T/\sigma_T^2 + I_d, \\ \boldsymbol{\Omega}_{t-1,t} &= -I_d, \quad t = 1, \ldots, T, \end{aligned}$$

where I_d is the $d \times d$ identity matrix and $\mathbf{c}_t$ is a column vector of dimension $d \times 1$, defined as

$$\mathbf{c}_0 = \mathbf{0}, \qquad \mathbf{c}_t = (\mathbf{F}_t^\top/\sigma_t^2)y_t^\star, \quad t = 1, \ldots, T.$$

In the homoscedastic case, $\sigma_1^2 = \ldots = \sigma_T^2 = \sigma^2$.

B. Multicore LPDS calculation

In the code below, the following R packages are used: **doParallel** (Microsoft and Weston 2020a) and **foreach** (Microsoft and Weston 2020b) for multicore computations, **zoo** (Zeileis and Grothendieck 2005; Zeileis *et al.* 2021) for manipulating dates, and **RhpcBLASctl** (Nakano and Nakama 2021) for controlling the number of BLAS threads.

```
R> library("doParallel")
R> library("foreach")
R> library("zoo")
R> library("RhpcBLASctl")
R> Tmax <- nrow(us_data) - 1
R> T0 <- Tmax - 49
R> ncores <- 4
R> cl <- makeCluster(ncores)
R> registerDoParallel(cl)
```

```
R> lpds <- foreach(t = T0:Tmax, .combine = "cbind",
+    .packages = c("RhpcBLASctl", "shrinkTVP"),
+    .errorhandling = "pass") %dopar% {
+    set.seed(t)
+    niter <- 30000
+    nburn <- 15000
+    nthin <- 5
+    blas_set_num_threads(1)
+
+    data_test <- us_data[t+1,]
+    data_t <- us_data[1:t,]
+
+    # Fully hierarchical triple gamma
+    res_FH_TG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      mod_type = "triple", niter = niter, nburn = nburn, nthin = nthin)
+
+    # Hierarchical triple gamma
+    res_H_TG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      mod_type = "triple", niter = niter, nburn = nburn, nthin = nthin,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      learn_c_xi = FALSE, learn_c_tau = FALSE)
+
+    # Non-hierarchical triple gamma
+    res_TG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      mod_type = "triple", niter = niter, nburn = nburn, nthin = nthin,
+      learn_kappa2_B = FALSE, learn_lambda2_B = FALSE,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      learn_c_xi = FALSE, learn_c_tau = FALSE)
+
+    # Hierarchical horseshoe
+    res_H_HS <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      mod_type = "triple", niter = niter, nburn = nburn, nthin = nthin,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      learn_c_xi = FALSE, learn_c_tau = FALSE,
+      a_xi = 0.5, a_tau = 0.5, c_xi = 0.5, c_tau = 0.5)
+
+    # Non-hierarchical horseshoe
+    res_HS <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      mod_type = "triple", niter = niter, nburn = nburn, nthin = nthin,
+      learn_kappa2_B = FALSE, learn_lambda2_B = FALSE,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      learn_c_xi = FALSE, learn_c_tau = FALSE,
+      a_xi = 0.5, a_tau = 0.5, c_xi = 0.5, c_tau = 0.5)
+
+    # Fully hierarchical double gamma
+    res_FH_DG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      niter = niter, nburn = nburn, nthin = nthin,
+       hyperprior_param = list(nu_tau = 1, nu_xi = 1))
+
+     # Hierarchical double gamma
+     res_H_DG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       niter = niter, nburn = nburn, nthin = nthin,
+       learn_a_xi = FALSE, learn_a_tau = FALSE,
+       a_xi = 0.1, a_tau = 0.1)
+
+     # Non-hierarchical double gamma
+     res_DG <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       niter = niter, nburn = nburn, nthin = nthin,
+       learn_a_xi = FALSE, learn_a_tau = FALSE,
+       a_xi = 0.1, a_tau = 0.1,
+       learn_kappa2_B = FALSE, learn_lambda2_B = FALSE)
+
+     # Hierarchical Lasso
+     res_H_LS <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       niter = niter, nburn = nburn, nthin = nthin,
+       learn_a_xi = FALSE, learn_a_tau = FALSE,
+       a_xi = 1, a_tau = 1)
+
+     # Non-hierarchical Lasso
+     res_LS <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       niter = niter, nburn = nburn, nthin = nthin,
+       learn_a_xi = FALSE, learn_a_tau = FALSE,
+       a_xi = 1, a_tau = 1,
+       learn_kappa2_B = FALSE, learn_lambda2_B = FALSE)
+
+     # Ridge regression
+     res_FV <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       mod_type = "ridge", niter = niter, nburn = nburn, nthin = nthin)
+
+     lpds_res <- c(LPDS(res_FH_TG, data_test),
+       LPDS(res_H_TG, data_test),
+       LPDS(res_TG, data_test),
+       LPDS(res_H_HS, data_test),
+       LPDS(res_HS, data_test),
+       LPDS(res_FH_DG, data_test),
+       LPDS(res_H_DG, data_test),
+       LPDS(res_DG, data_test),
+       LPDS(res_H_LS, data_test),
+       LPDS(res_LS, data_test),
+       LPDS(res_FV, data_test))
+     rm(list = ls()[!ls() %in% c("lpds_res", "us_data")])
+     return(lpds_res)
+   }
R> stopCluster(cl)
```

Plot the results:

```
R> cumu_lpds <- apply(lpds, 1, cumsum)
R> color <- c(rep("cyan3", 3), rep("firebrick3", 2), rep("forestgreen", 3),
+    rep("yellow2", 2), "black")
R> lty <- c(1:3, 1:2, 1:3, 1:2, 1)
R> par(mar = c(6, 4, 1, 1))
R> colnames(cumu_lpds) <- c("Fully hierarchical NGG", "Hierarchical NGG",
+    "NGG", "Hierarchical Horseshoe", "Horseshoe", "Fully hierarchical NG",
+    "Hierarchical NG", "NG", "Hierarchical Lasso", "Lasso",
+    "Ridge Regression")
R> matplot(cumu_lpds, type = "l", ylab = "Cumulative LPDS",
+    xaxt = "n", xlab = "", col = color, lty = lty, lwd = 1.5)
R> labs <- as.yearmon(time(usmacro.update))[T0:Tmax + 1][c(FALSE, TRUE)]
R> axis(1, at = (1:length(T0:Tmax))[c(FALSE, TRUE)], labels = FALSE)
R> text(x = (1:length(T0:Tmax))[c(FALSE, TRUE)],
+    y = par()$usr[3] - 0.05 * (par()$usr[4] - par()$usr[3]),
+    labels = labs, srt = 45, adj = 1, xpd = TRUE)
R> legend("topright", colnames(cumu_lpds), col = color,
+    lty = lty, lwd = 1.5, bty = "n", cex = 0.8)
```

Affiliation:

Peter Knaus
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
WU Vienna University of Economics and Business
Welthandelsplatz 1, Building D4, Entrance A, 3rd Floor
1020 Vienna, Austria
E-mail: peter.knaus@wu.ac.at
URL: https://www.wu.ac.at/statmath/faculty-staff/faculty/peter-knaus

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
November 2021, Volume 100, Issue 13 *Submitted:* 2019-07-12
doi:10.18637/jss.v100.i13 *Accepted:* 2021-02-26