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

Angela Bitto-Nemling **Annalisa Cadonna** WU Vienna **Sylvia Frühwirth-Schnatter** WU Vienna

Peter Knaus
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 2019), which provides a fully Bayesian implementation of shrinkage priors for TVP models, taking advantage of recent developments in the literature, in particular that of Bitto and Frühwirth-Schnatter (2019). The package **shrinkTVP** allows for posterior simulation of the parameters through an efficient Markov Chain Monte Carlo (MCMC) scheme. Moreover, summary and visualization methods, as well as the possibility of assessing predictive performance through log predictive density scores (LPDSs), 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) and Bitto and Frühwirth-Schnatter (2019). 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 (2017).

In this paper, we describe the **shrinkTVP** package (Knaus *et al.* 2019) 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/web/packages/shrinkTVP`. The package efficiently implements recent developments in the Bayesian literature, in particular the ones presented in Bitto and Frühwirth-Schnatter (2019). The computationally intensive algorithms in the package are written in C++ and interfaced with R (R Core Team 2017) via the **Rcpp** (Eddelbuettel and Balamuta 2017) 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 goal is 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. Coupled with intuitive summary and plot methods, this leads to a package that is both easy to use while remaining highly flexible.

shrinkTVP is, to our knowledge, the only R package that combines TVP models with shrinkage priors. In the TVP models context, the most popular R package is **dlm** (Petris 2010), which provides routines for maximum likelihood estimation, Kalman filtering and smoothing, and Bayesian analysis for dynamic linear models (DLMs), of which TVP models are a subset. The package **bsts** (Scott 2019) performs Bayesian analysis for the closely related class of structural time series models. Moreover, a number of R packages providing regularization and shrinkage methods are available. For example, **shrink** (Dunkler, Sauerbrei, and Heinze 2016) implements various shrinkage methods for linear, generalized linear, or Cox regressions, **biglasso** (Zeng and Breheny 2017) aims at very fast lasso-type models for high-dimensional linear regression and **glmnet** (Friedman, Hastie, and Tibshirani 2010) provides efficient procedures for implementing elastic-net regularization for a variety of models. With regards to Bayesian shrinkage, the normal-beta prime shrinkage prior is implemented in the package **NormalBetaPrime** (Bai and Ghosh 2019) and the popular horseshoe prior in the package **horseshoe** (van der Pas, Scott, Chakraborty, and Bhattacharya 2016). Both of these packages focus on high-dimensional regression models, and do not provide shrinkage for TVP models.

The remainder of the paper is organized as follows. Section 2 briefly introduces TVP models and the normal-gamma shrinkage priors, and describes the Markov Chain Monte Carlo (MCMC) algorithm 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 return object of `shrinkTVP`. Section 4 explains how to assess the performance of the model 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 from the **bvarsv** (Krueger 2015) package. 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

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

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,

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

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 priors (Griffin and Brown 2010) 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 double gamma prior on the innovation variances θ_j, for $j = 1, \dots, d$. This prior can be represented as a conditionally normal distribution, with the component specific variance following a gamma distribution, that is

$$\sqrt{\theta}_j | \xi_j^2 \sim \mathcal{N}\left(0, \xi_j^2\right), \qquad \xi_j^2 | a^\xi, \kappa^2 \sim \mathcal{G}\left(a^\xi, \frac{a^\xi \kappa^2}{2}\right), \tag{4}$$

$$\beta_j | \tau_j^2 \sim \mathcal{N}\left(0, \tau_j^2\right), \qquad \tau_j^2 | a^\tau, \lambda^2 \sim \mathcal{G}\left(a^\tau, \frac{a^\tau \lambda^2}{2}\right). \tag{5}$$

The prior variances ξ_j^2 and τ_j^2 are referred to as *local shrinkage parameters*, in that they control the strength with which each individual parameter $\sqrt{\theta}_j$ and β_j is pulled toward zero. The parameters κ^2 and λ^2 are dubbed *global shrinkage parameters*, as they determine how strongly *all* parameters are pulled to zero. Since $\mathrm{E}(\theta_j|a^\xi, \kappa^2) = 2/\kappa^2$ and $\mathrm{E}(\beta_j^2|a^\tau, \lambda^2) = 2/\lambda^2$, the larger κ^2 and λ^2, the stronger this effect. Finally, we refer to a^ξ and a^τ as *shrinkage adaption parameters*. As a^ξ and a^τ decrease, marginally more mass is placed around zero and jointly more mass is put on sparse specifications of the model. In particular, setting the local adaption parameters, a^ξ and a^τ, equal to one results in a Bayesian Lasso (Park and Casella 2008) prior on the $\sqrt{\theta}_j$'s and the β_j's, respectively.

The parameters κ^2, λ^2, a^ξ, a^τ can be learned from the data through appropriate prior distributions. As priors for the global shrinkage parameters, we use

$$\kappa^2 \sim \mathcal{G}(d_1, d_2), \qquad \lambda^2 \sim \mathcal{G}(e_1, e_2). \tag{6}$$

Moreover, in order to learn the shrinkage adaption parameters, we generalize the approach taken in Bitto and Frühwirth-Schnatter (2019) and place the following gamma distributions as priors:

$$a^\xi \sim \mathcal{G}(\nu^\xi, \nu^\xi b^\xi), \qquad a^\tau \sim \mathcal{G}(\nu^\tau, \nu^\tau b^\tau), \tag{7}$$

which corresponds to the exponential prior used in Bitto and Frühwirth-Schnatter (2019) when $\nu^\xi = 1$ and $\nu^\tau = 1$. The parameters ν^ξ and ν^τ 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,

$$\sigma^2|C_0 \sim \mathcal{G}^{-1}(c_0, C_0), \qquad C_0 \sim \mathcal{G}(g_0, G_0), \tag{8}$$

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

$$\mu \sim \mathcal{N}(b_\mu, B_\mu), \quad \frac{\phi + 1}{2} \sim \mathcal{B}(a_\phi, b_\phi), \quad \sigma_\eta^2 \sim \mathcal{G}(1/2, 1/2B_\sigma), \tag{9}$$

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-Hasting 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) for further details.

Algorithm 1 *Gibbs Sampling Algorithm*

1. *Sample the latent states* $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_0, \ldots, \tilde{\boldsymbol{\beta}}_T)$ *in the non-centered parametrization from a multivariate normal distribution;*

2. *Sample jointly* $\beta_1, \ldots, \beta_d$, *and* $\sqrt{\theta_1}, \ldots, \sqrt{\theta_d}$ *in the non-centered parametrization from a multivariate normal distribution;*

3. *Perform an ancillarity-sufficiency interweaving step and redraw* $\beta_1, \ldots, \beta_d$, *each from a normal distribution and* $\theta_1, \ldots, \theta_d$, *each from a generalized inverse Gaussian distribution using* **GIGrvg** *(Hörmann and Leydold 2015);*

4. (a) *Sample the variance shrinkage adaption parameter* a^ξ *using a random walk Metropolis-Hastings step;*
   (b) *Sample the parameter shrinkage adaption parameter* a^τ *using a random walk Metropolis-Hastings step;*

5. (a) *Sample the local shrinkage parameters* ξ_j^2, *for* $j = 1, \ldots, d$, *from conditionally independent generalized inverse Gaussian distributions;*
   (b) *Sample the local shrinkage prameters* τ_j^2, *for* $j = 1, \ldots, d$, *from conditionally independent generalized inverse Gaussian distributions;*

6. *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 latent log-volatilities* $\boldsymbol{h} = (h_0, \ldots, h_T)$ *using* **stochvol** *(Kastner 2016).*

When fitting the model under the full hierarchical shrinkage prior defined in Equations (4)–(7), all steps in Algorithm 1 are performed, while steps 4(a), 4(b), 5(a) and 5(b) are skipped in certain prior setups.

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 the precision matrix of the full conditional distribution of $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_0, \ldots, \tilde{\boldsymbol{\beta}}_T)$, 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).

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 priors (4) to (7) and the following choice for the hyperparameters: $d_1 = d_2 = e_1 = e_2 = 0.001$, $\nu^\xi = \nu^\tau = 5$ and $b^\xi = b^\tau = 10$, 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 (8) 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 2017)). 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>
R> # Baseline model
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): 4.39 seconds.
3417 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** (Plummer, Best, Cowles, and Vines 2006) compatible, so that the user can utilize the tools provided by the excellent package **coda** to assess convergence.

3.2. Specifying the priors

	Shrinkage on $\sqrt{\theta_j}$		Shrinkage on β_j	
	a^ξ	κ^2	a^τ	λ^2
Full hierarchical shrinkage prior	$\mathcal{G}(\nu^\xi, \nu^\xi b^\xi)$	$\mathcal{G}(d_1, d_2)$	$\mathcal{G}(\nu^\tau, \nu^\tau b^\tau)$	$\mathcal{G}(e_1, e_2)$
Hierarchical normal-gamma prior	fixed	$\mathcal{G}(d_1, d_2)$	fixed	$\mathcal{G}(e_1, e_2)$
Normal-gamma prior	fixed	fixed	fixed	fixed
Bayesian Lasso	fixed at 1	fixed	fixed at 1	fixed

Table 1: Overview of different possible model specifications

More granular control over the prior setup can be exercised by specifying all or a subset of the parameters. In addition to changing the hyperparameters given in Section 2.2, the user can fix one or both values of the global shrinkage parameters (κ^2, λ^2) and the shrinkage adaption parameters (a^τ, a^ξ), instead of learning them from the data as done in the default setting. The benefit of this 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. For example, setting the local adaption parameters, a^ξ and a^τ, equal to one results in a Bayesian Lasso (Park and Casella 2008) 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 shrinkage adaption parameters to a value closer to zero. Table 1 gives an overview of different model specifications. Note that separate prior choices can be made 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. 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 shrinkage adaption parameters It is possible to set the shrinkage adaption 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 $\sqrt{\theta_j}$ and on the β_j:

```
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 $\kappa^2(\lambda^2)$ by specifying the argument `kappa2` (`lambda2`), after setting `learn_k2` (`learn_lambda2`) to `FALSE`. In the code below, we give an example on how to fit a (non-hierarchical) Bayesian Lasso on both $\sqrt{\theta_j}$ and β_j, 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 = FALSE, learn_lambda2 = FALSE, kappa2 = 100, lambda2 = 100,
+    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 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 9, 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 prior distributions. This can be done through the arguments `hyperprior_param` and `sv_param`, which both have to be named lists.

In addition to user defined hyperparameters, unspecified parameters will be set to default values, as defined in Section 3.2.

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

3.5. Posterior inference: Summarize and visualize the posterior distribution

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

1. the parameter draws of σ^2 in `sigma2`,
2. the parameter draws of $(\sqrt{\theta_1}, \ldots, \sqrt{\theta_d})$ in `theta_sr`,
3. the parameter draws of $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$ in `beta_mean`,
4. a list holding d `mcmc.tvp` objects (one for each $\boldsymbol{\beta}_j = (\beta_{j0}, \ldots, \beta_{jT})$) containing the parameter draws in `beta`,
5. the parameter draws of $\xi_1^2, \ldots, \xi_d^2$, in `xi2`,
6. the parameter draws of a^ξ in `a_xi`,
7. some acceptance statistics for the Metropolis Hastings step for a^ξ in `a_xi_acceptance`,
8. the parameter draws of $\tau_1^2, \ldots, \tau_d^2$ in `tau2`,

9. the parameter draws of a^τ in `a_tau`,
10. some acceptance statistics for the Metropolis Hastings step for a^τ in `a_tau_acceptance`,
11. the parameter draws for κ^2 in `kappa2`,
12. the parameter draws for λ^2 in `lambda2`,
13. the parameter draws of C_0 in `C0`,
14. the prior hyperparameters in `priorvals`,
15. the design matrix and the response in `model`, and
16. summary statistics for the parameter draws in `summaries`.

When some parameters are fixed by the user, the corresponding output value is omitted. 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_res` objects. `summary` has two arguments beyond the mandatory `shrinkTVP_res` 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
 sigma2                   1.018   0.11     1.013  0.815    1.239     3343.207

 abs(theta_sr_Intercept) 0.198   0.046    0.193  0.118    0.289     268.578
 abs(theta_sr_x1)         0.007   0.015    0.001  0        0.04      284.642
 abs(theta_sr_x2)         0.004   0.008    0      0        0.018     610.114

 beta_mean_Intercept      1.067   0.67     1.141  -0.08    2.144     229.597
 beta_mean_x1             -0.297  0.126    -0.308 -0.491   0.013     486.943
 beta_mean_x2             0.001   0.037    0      -0.089   0.083     3643.73

 xi2_Intercept            5.924   102.961  0.06   0.001    2.869     3979.303
 xi2_x1                   0.044   1.336    0      0        0.018     5000
 xi2_x2                   0.073   1.683    0      0        0.005     5000

 a_xi                     0.09    0.04     0.083  0.031    0.175     465.036
```

```
tau2_Intercept      161.552 4343.276 1.447  0        79.21     5000
tau2_x1             32.77   1105.99  0.184  0        11.658    5000
tau2_x2             0.451   10.085   0      0        0.107     5000

a_tau               0.099   0.042    0.092  0.027    0.183     574.134

kappa2              75.682  204.592  6.363  0        398.87    3932.451

lambda2             9.399   42.184   0.317  0        41.283    823.744

C0                  1.721   0.629    1.644  0.651    2.983     5008.619
```

The `plot` method can be used to visualize the posterior distribution estimated by `shrinkTVP`. Aside from the `shrinkTVP_res` object, its arguments are the mandatory `pars`, a character vector containing the names of the parameters to visualize, and in the case of time-varying parameters `nplot`, which controls the number of plots to display per page. The names supplied in `pars` have to coincide with the names of the list elements of the `shrinkTVP_res` object. The default value is `c("beta")`, i.e. empirical quantiles of the posterior of $\boldsymbol{\beta}_t$ over time are shown. If there are too many plots to fit on one page, `plot` will cycle through all parameters to display. It will call either `plot.mcmc` from the **coda** package, if the parameter is non time-varying, or `plot.mcmc.tvp` from the **shrinkTVP** package for time-varying parameters, passing all additional arguments specified via `...` to the respective plotting functions. See the code below for an example and Figure 1 for the corresponding output.

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

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_res` object. An example of this can be seen in Figure 2, where `pars = "theta_sr"`, changes the output to a graphical summary of the parameter draws of $\sqrt{\theta_1}, \ldots, \sqrt{\theta_d}$, using **coda**'s `plot.mcmc` function. To obtain Figure 2, one can run

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

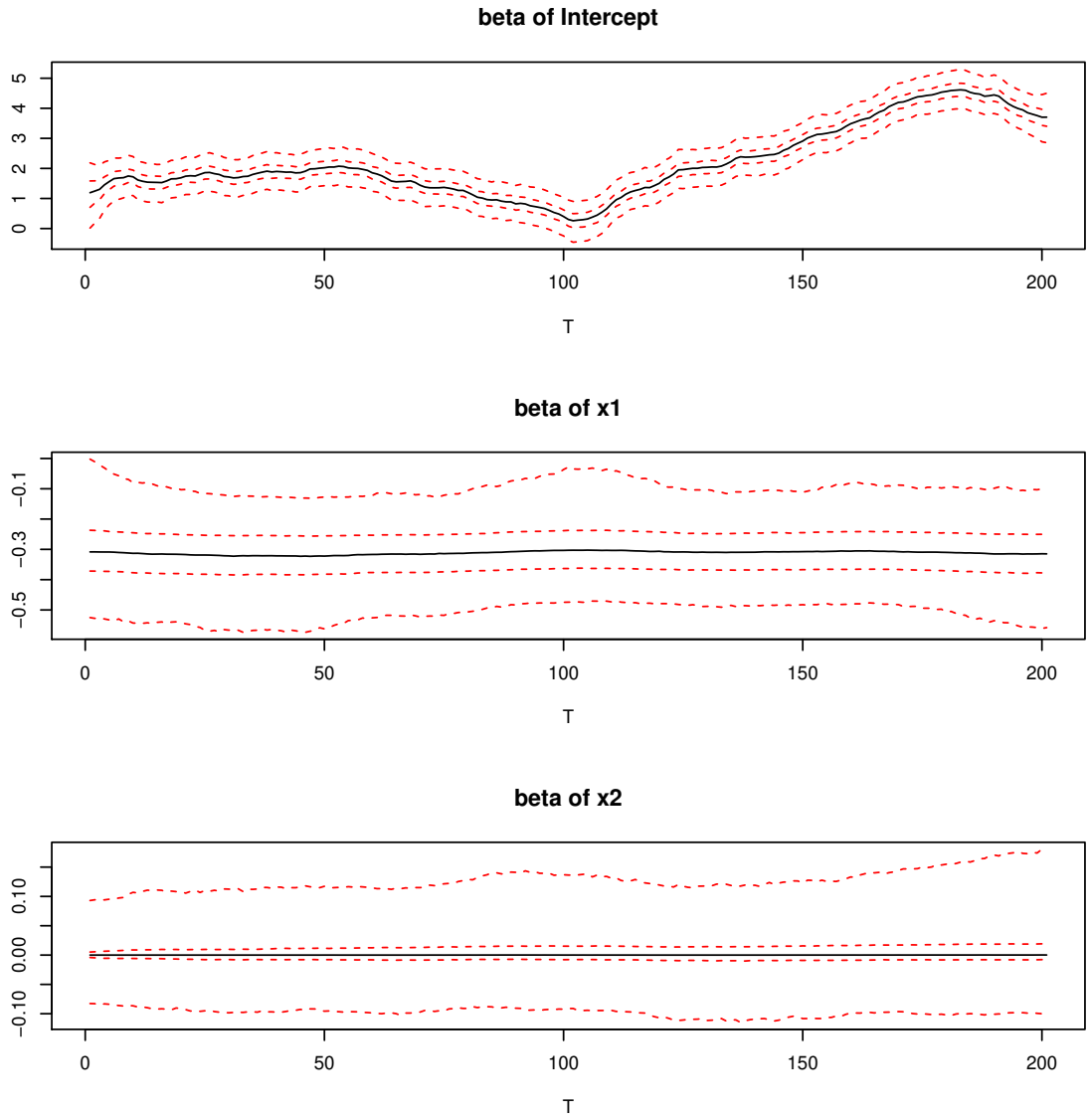

Figure 1: Visualization of the evolution of the time-varying parameter $\boldsymbol{\beta}_j = (\beta_{j0}, \ldots, \beta_{jT}), j = 1, \ldots, 3$, over time $t = 0, \ldots, T$, as provided by the `plot` method. `plot` is in turn calling `plot.mcmc.tvp` on the individual `mcmc.tvp` objects. The median is always displayed as a black line, and the red dotted lines indicate the pointwise 2.5%, 25%, 75% and 97.5% quantiles, unless otherwise specified.

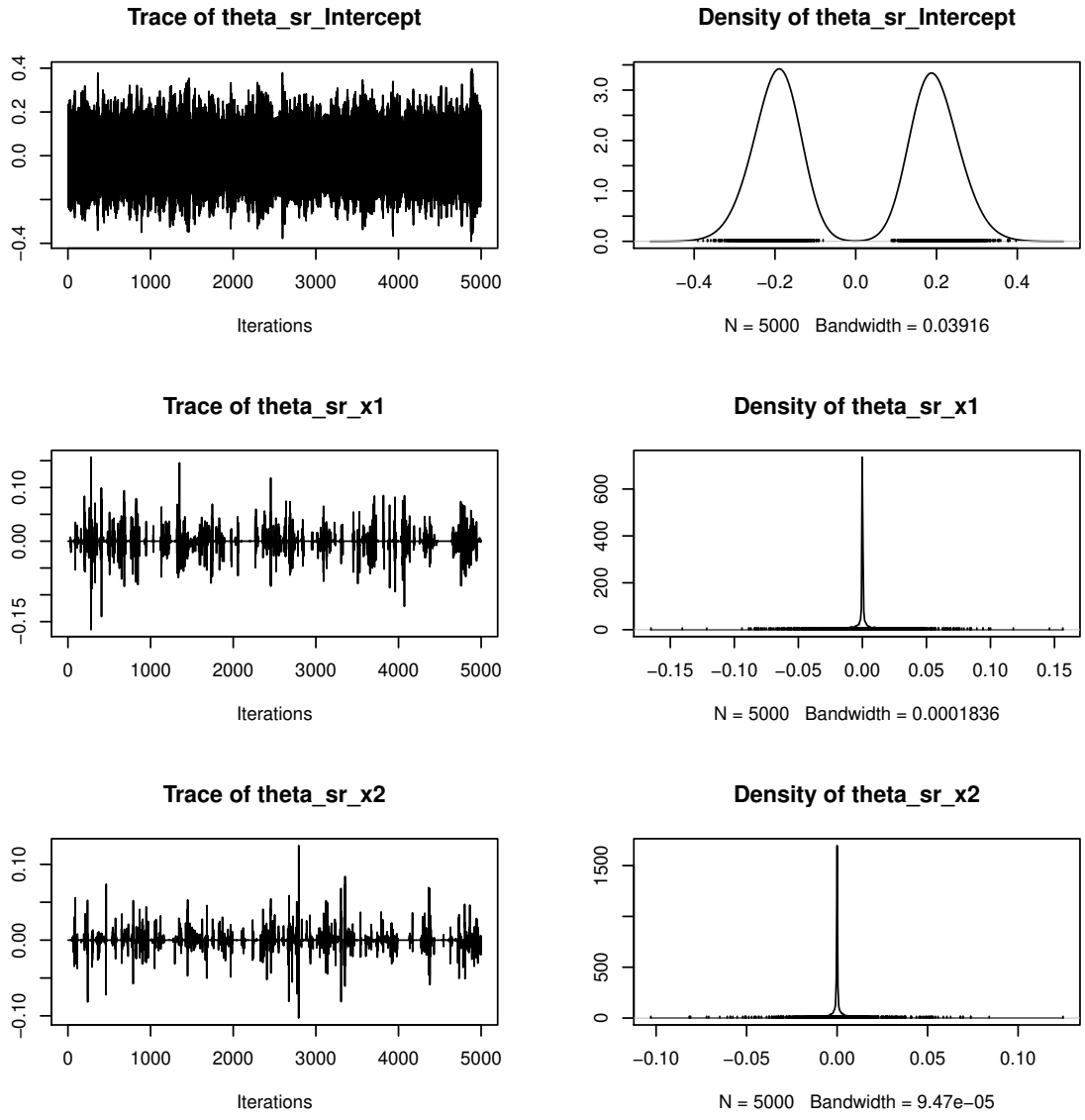

Figure 2: Trace plots (left column) and kernel density estimates of the posterior density (right column) for the parameters $\sqrt{\theta}_1, \ldots, \sqrt{\theta}_3$, as provided by the `plot` method. `plot` is in turn calling **coda**'s `plot.mcmc`.

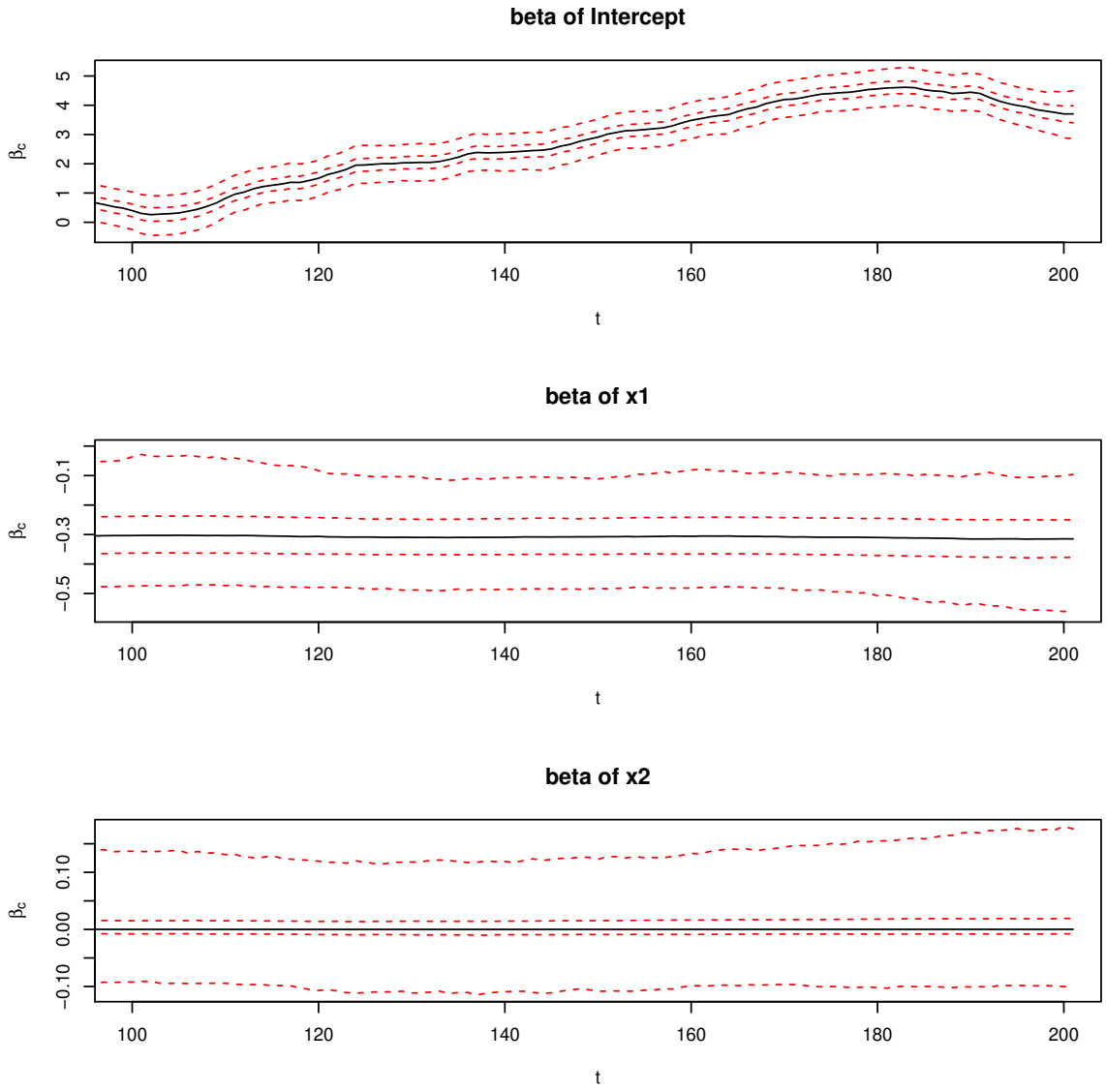

Figure 3: Visualization of the evolution of the time-varying parameter β_{jt} over time $t = 100, \ldots, 200$ for $j = 1, \ldots, 3$, as provided by the `plot` method. `plot` is in turn calling `plot.mcmc.tvp` on the individual `mcmc.tvp` objects. Arbitrary arguments can be passed to `plot`, in this example the x-axis of the plot was restricted with `xlim` and the label of the y-axis was changed with `ylab`. The median is always displayed as a black line, and the red dotted lines indicate the pointwise 2.5%, 25%, 75% and 97.5% quantiles, unless otherwise specified.

The `plot` method will pass any additional arguments on to the respective plotting functions, allowing for some flexibility in the displayed plots. An example of this behavior is demonstrated in the code below and the resulting Figure 3, where the x-axis is truncated with the `xlim` argument and the label for the y-axis is changed by the `ylab` argument.

```
R> plot(res, pars = "beta", xlim = c(100, 200),
+    ylab = expression(beta[c]), xlab = "t")
```

The `plot.mcmc.tvp` method displays empirical posterior quantiles of a single time-varying parameter over time. Instead of being called indirectly via `plot`, the user can also directly call it by calling `plot` on a single `mcmc.tvp` object within the `shrinkTVP_res` object. `plot.mcmc.tvp` takes one additional mandatory argument, `probs`, which is a vector of quantiles to plot. `plot.mcmc.tvp` will automatically add 0.5, i.e. the median, to `probs` if the user does not. It will pass on any additional arguments specified via `...`, allowing the user some flexibility concerning the final plot. In the code below, this behavior is exploited to add a title to the plot, resulting in Figure 4.

```
R> par(cex.main = 1, mar = c(3, 4, 4, 2) + 0.1)
R> plot(res_sv$sigma2, xlab = "",
+   main = bquote("Traceplot of posterior of " ~ sigma[t]^2))
```

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

$$p(y_{t_0+1}|y_1, \ldots, y_{t_0}, \boldsymbol{x}_{t_0+1}) = \int p(y_{t_0+1}|\boldsymbol{x}_{t_0+1}, \boldsymbol{\psi}) p(\boldsymbol{\psi}|y_1, \ldots, y_{t_0}) d\boldsymbol{\psi}, \tag{10}$$

where $\boldsymbol{\psi}$ is the set of model parameters and latent variables up to $t_0 + 1$. For a TVP model with homoscedastic errors, $\boldsymbol{\psi} = (\tilde{\boldsymbol{\beta}}_0, \ldots \tilde{\boldsymbol{\beta}}_{t_0+1}, \sqrt{\theta}_1, \ldots, \sqrt{\theta_d}, \beta_1, \ldots, \beta_d, \sigma^2)$, whereas for a TVP model with SV errors, $\boldsymbol{\psi} = (\tilde{\boldsymbol{\beta}}_0, \ldots \tilde{\boldsymbol{\beta}}_{t_0+1}, \sqrt{\theta}_1, \ldots, \sqrt{\theta_d}, \beta_1, \ldots, \beta_d, \sigma_1^2, \ldots, \sigma_{t_0+1}^2)$. Given M samples from the posterior distribution of the parameters and latent variables, $p(\boldsymbol{\psi}|y_1, \ldots, y_{t_0})$, Monte Carlo integration could be applied immediately to approximate (10). 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, the predictive density approximated by Monte Carlo integration is given by

$$p(y_{t_0+1}|y_1, \ldots, y_{t_0}, \boldsymbol{x}_{t_0+1}) \approx \frac{1}{M} \sum_{m=1}^{M} p(y_{t_0+1}|\boldsymbol{x}_{t_0+1}\boldsymbol{\beta}^{(m)} + \boldsymbol{F}_{t_0+1}^{(m)} \tilde{\boldsymbol{\beta}}_{t_0}^{(m)} + \boldsymbol{m}_{t_0}^{(m)}, \boldsymbol{F}_{t_0+1}^{(m)} (\boldsymbol{\Sigma}_{t_0}^{(m)} + I_d)(\boldsymbol{F}_{t_0+1}^{(m)})^\top + (\sigma^2)^{(m)}),$$

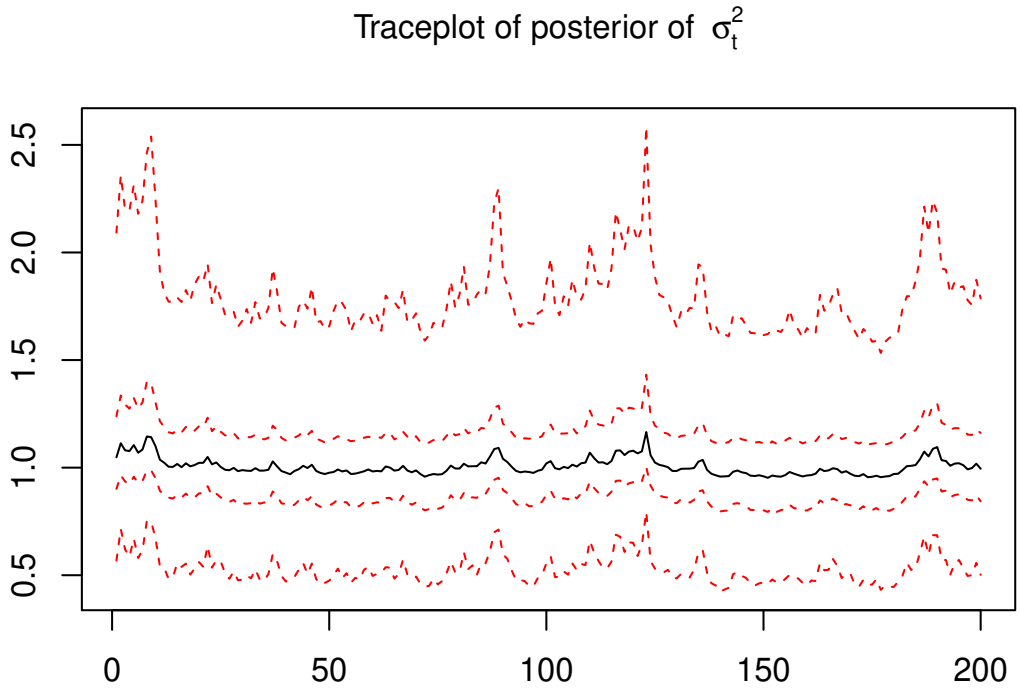

Figure 4: Visualization of the evolution of the stochastic volatility σ_t^2 over time $t = 1, \ldots, T$, as provided by the `plot.mcmc.tvp` method. The median is always displayed as a black line, and the red dotted lines indicate the pointwise 2.5%, 25%, 75% and 97.5% quantiles, unless otherwise specified.

where the conditional predictive densities of y_{t_0+1} are Gaussian with the mean and the variance depending on the MCMC draws. These moments are computed for the mth MCMC iteration from $\boldsymbol{F}_{t_0+1} = \boldsymbol{x}_{t_0+1}\text{Diag}(\sqrt{\theta_1}, \ldots, \sqrt{\theta_d})$ 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):

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

The quantities $\boldsymbol{c}_t$, $\boldsymbol{\Omega}_{tt}$ and $\boldsymbol{\Omega}_{t-1,t}$ for $t = 1, \ldots, t_0$ 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 $\sigma_{t_0+1}^2$, however it is not possible to integrate the likelihood with respect to both latent variables $\tilde{\boldsymbol{\beta}}_{t_0+1}$ and $\sigma_{t_0+1}^2$. Hence, at each MCMC iteration a draw is taken from the predictive distribution of $\sigma_{t_0+1}^2 = \exp(h_{t_0+1})$, 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 $(\mu, \phi, \sigma_\eta^2)^{(m)}$ and $h_{t_0}^{(m)}$, obtain a draw of $(\sigma_{t_0+1}^2)^{(m)}$;

2. calculate the conditionally optimal Kalman mixture approximation as

$$p(y_{t_0+1}|y_1,\ldots,y_{t_0},\boldsymbol{x}_{t_0+1}) \approx \frac{1}{M}\sum_{m=1}^{M} p(y_{t_0+1}|\boldsymbol{x}_{t_0+1}\boldsymbol{\beta}^{(m)} + \boldsymbol{F}_{t_0+1}^{(m)}\tilde{\boldsymbol{\beta}}_{t_0}^{(m)} + \boldsymbol{m}_{t_0}^{(m)}, \boldsymbol{F}_{t_0+1}^{(m)}(\boldsymbol{\Sigma}_{t_0}^{(m)} + I_d)(\boldsymbol{F}_{t_0+1}^{(m)})^\top + (\sigma_{t_0+1}^2)^{(m)}),$$

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

The **shrinkTVP** package provides a way to calculate the LPDSs using the function `shrinkTVP`. When `LPDS` is set to `TRUE`, the data provided via `data` or the formula interface are assumed to contain the covariates and response up to time t_0. The values of $\boldsymbol{x}_{t_0+1}$ are then supplied through `x_test`, while y_{t_0+1} is passed to `y_test`, as shown in the following snippet of code.

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

        [,1]
[1,] -1.398314
attr(,"type")
[1] "stat"
```

This leads to an additional output in the resulting `shrinkTVP_res` object called `LPDS`, which contains the calculated log predictive density score. For an example on how to calculate LPDSs for k points in time, please see Section 5.

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 (Krueger 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")
R> # Create matrix of lags and create final data set
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 run the default TVP model with the full hierarchical shrinkage 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)

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
sigma2                   0.019   0.006     0.018  0.008    0.03      1732.763

abs(theta_sr_Intercept) 0.141   0.024     0.142  0.092    0.186     716.452
abs(theta_sr_inf_lag)    0.043   0.006     0.043  0.03     0.056     2173.311
abs(theta_sr_une_lag)    0.004   0.006     0.001  0        0.016     80.178
abs(theta_sr_tbi_lag)    0.001   0.002     0      0        0.006     423.844

beta_mean_Intercept      0.352   0.411     0.196  -0.109   1.183     545.061
beta_mean_inf_lag        0.746   0.181     0.756  0.361    1.072     1072.752
beta_mean_une_lag        -0.127  0.07      -0.138 -0.233   0.004     102.591
beta_mean_tbi_lag        0.009   0.022     0      -0.023   0.067     590.672

xi2_Intercept            1.785   53.178    0.03   0.001    0.962     5000
xi2_inf_lag              0.238   3.953     0.005  0        0.216     5000
xi2_une_lag              0.018   0.427     0      0        0.007     5000
xi2_tbi_lag              0.024   1.126     0      0        0.001     5000

a_xi                     0.094   0.041     0.089  0.022    0.172     548.879

tau2_Intercept           225.777 12323.864 0.081  0        10.82     5000
tau2_inf_lag             18.332  377.616   0.765  0        20.478    5000
tau2_une_lag             2.08    31.701    0.037  0        3.022     5000
tau2_tbi_lag             0.084   1.113     0      0        0.066     5000

a_tau                    0.1     0.044     0.094  0.026    0.183     754.434

kappa2                   118.658 252.723   21.335 0        576.182   4745.271
```

lambda2	8.363	28.274	1.025	0	36.827	5000
C0	0.133	0.062	0.121	0.028	0.254	3119.826

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 $p(\sqrt{\theta_j}|\mathbf{y})$ 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 five different prior choices: (1) the full hierarchical shrinkage prior, (2) the hierarchical normal-gamma prior with fixed $a^\xi = a^\tau = 0.1$, (3) the normal-gamma prior with $a^\xi = a^\tau = 0.1$ and $\kappa^2 = \lambda^2 = 20$, (4) the hierarchical Bayesian Lasso, and (5) the Bayesian Lasso with $\kappa^2 = \lambda^2 = 20$. Figure 7 shows the cumulative LPDSs for the last 50 quarters of the `usmacro.update` dataset. The default prior, that is the fully hierarchical shrinkage prior on both the β_j's and the $\sqrt{\theta_j}$'s, performs the best in terms of prediction amongst the five fitted models. In Appendix B we show how to obtain LPDSs for different models and points in time, using the packages **foreach** (Microsoft and Weston 2017) and **doParallel** (Microsoft and Weston 2018).

6. Conclusions

The goal of this paper was to introduce the reader to the functionality of the R package **shrinkTVP** (Knaus *et al.* 2019). 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 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 and use log-predictive density scores to compare various model specifications.

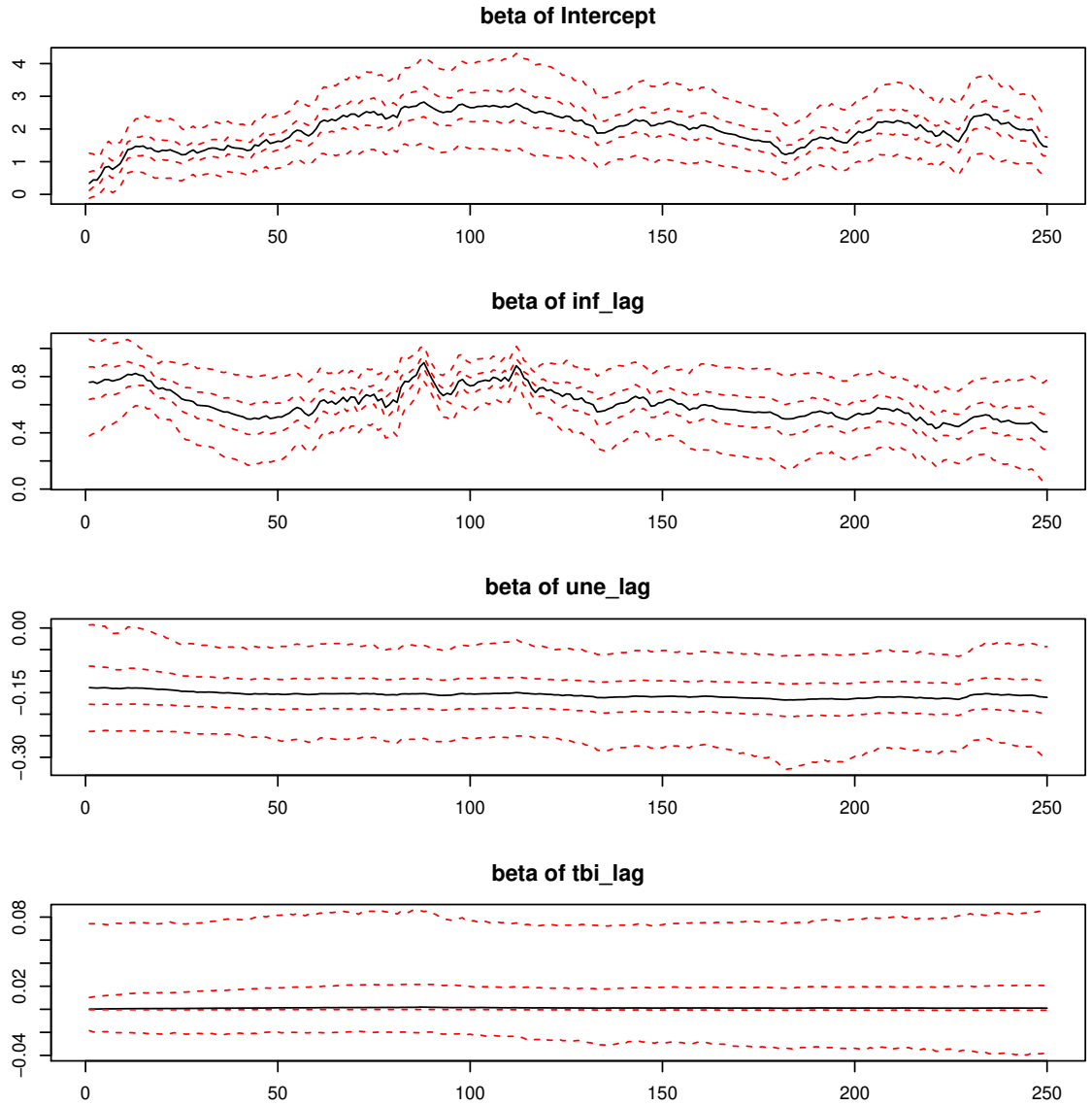

Figure 5: Visualization of the evolution of the time-varying parameter $\boldsymbol{\beta}_j = (\beta_{j0}, \ldots, \beta_{jT})$ over time $t = 0, \ldots, T$ for $j = 1, \ldots, 4$ for the `usmacro.update` dataset. The median is displayed as a black line, and the red dotted lines indicate the pointwise 2.5%, 25%, 75% and 97.5% quantiles.

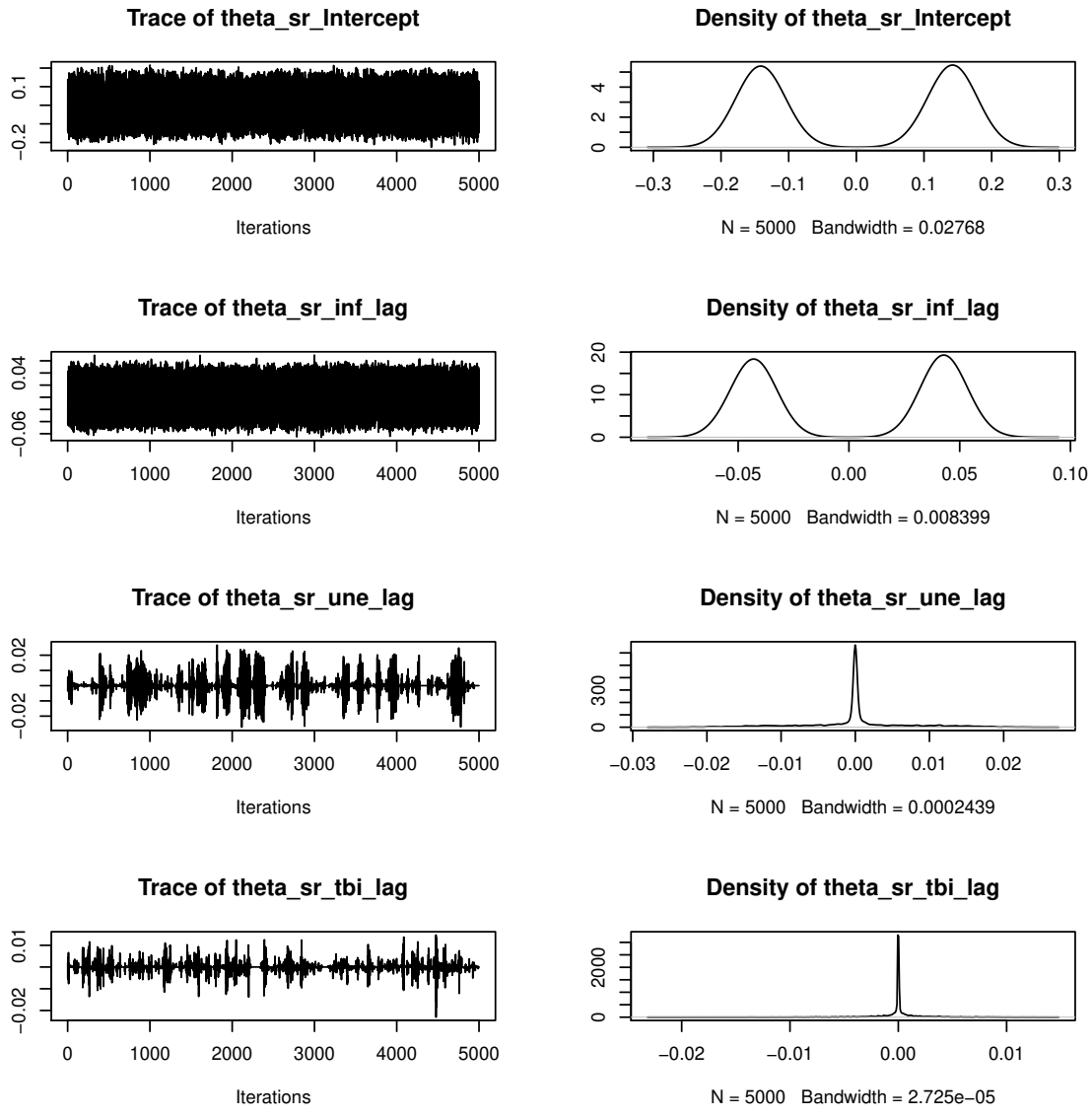

Figure 6: Trace plots (left column) and kernel density estimates of the posterior density (right column) for the parameters $\sqrt{\theta_1}, \ldots, \sqrt{\theta_4}$ for the `usmacro.update` dataset.

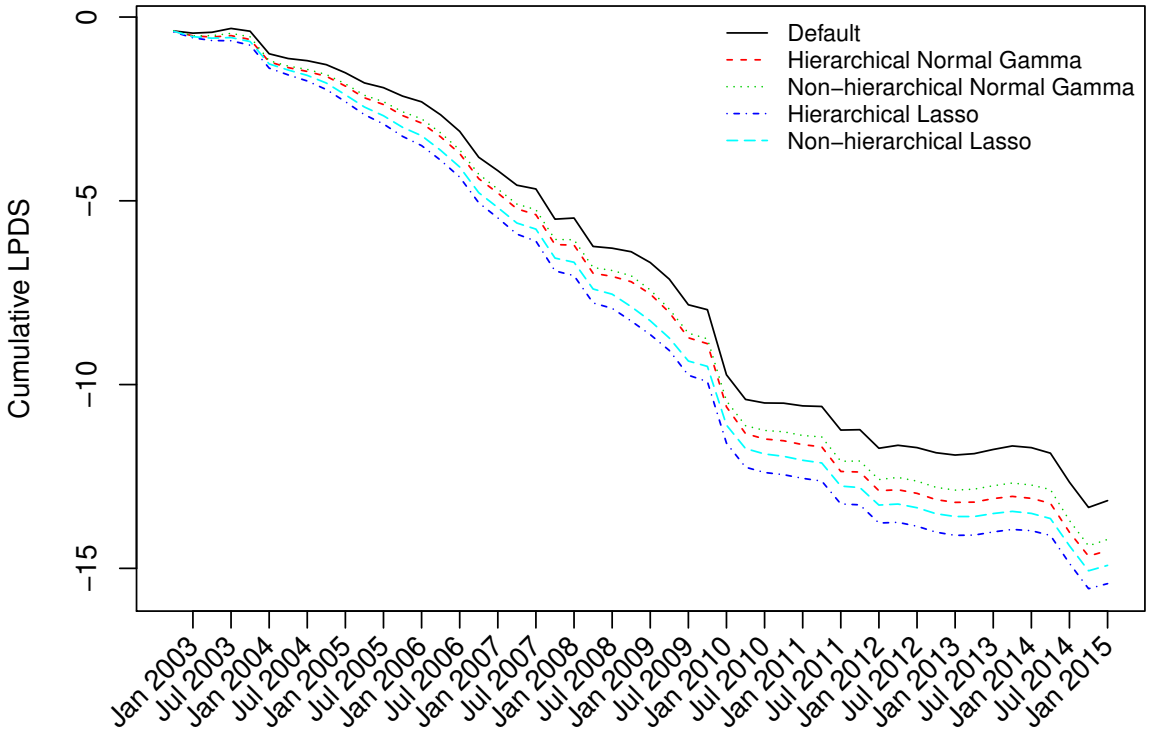

Figure 7: Cumulative LPDSs scores for the last 50 quarters of the `usmacro.update` dataset, for five different shrinkage priors: (1) the full hierarchical shrinkage prior, (2) the hierarchical normal-gamma prior with fixed $a^\xi = a^\tau = 0.1$, (3) the normal-gamma prior with $a^\xi = a^\tau = 0.1$ and $\kappa^2 = \lambda^2 = 20$, (4) the hierarchical Bayesian Lasso, and (5) the Bayesian Lasso with $\kappa^2 = \lambda^2 = 20$.

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.

A plan for further versions of the package is to implement additional shrinkage priors for TVP models such as the well-known horseshoe prior (Bhadra *et al.* 2017).

A. Appendix: 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):

$$\tilde{\boldsymbol{\beta}} | \boldsymbol{\beta}, \mathbf{Q}, \sigma_1^2, \ldots, \sigma_T^2, y_1^\star, \ldots y_T^\star \sim \mathcal{N}_{(T+1)d}\left(\boldsymbol{\Omega}^{-1}\mathbf{c}, \boldsymbol{\Omega}^{-1}\right) \tag{11}$$

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 Core Team 2017) packages are used: **doParallel** (Microsoft and Weston 2018), **foreach** (Microsoft and Weston 2017), **zoo** (Zeileis and Grothendieck 2005), and **RhpcBLASctl** (Nakano and Nakama 2018).

```
R> # Calculate LPDS in multicore
R> # Load libraries for multicore computations
R> library(doParallel)
R> library(foreach)
R>
R> # For manipulating dates
R> library(zoo)
R>
R> # Load library for controlling number of BLAS threads
R> library(RhpcBLASctl)
```

```
R>
R> # Define how many periods to calculate LPDS for
R> Tmax <- nrow(us_data) - 1
R> T0 <- Tmax - 49
R>
R> # Determine number of cores to be used and register parallel backend
R> ncores <- 4
R> cl <- makeCluster(ncores)
R> registerDoParallel(cl)
R>
R> lpds <- foreach(t = T0:Tmax, .combine = "cbind",
+    .packages = c("RhpcBLASctl", "shrinkTVP")) %dopar% {
+
+    # Set number of BLAS threads, so they dont interfere with each other
+    blas_set_num_threads(1)
+
+    # Create data_t from all data up to time t and
+    # y_test and x_test from data at time t+1
+    y_test <- us_data[t+1, "inf"]
+    x_test <- us_data[t+1, c("inf_lag", "une_lag", "tbi_lag")]
+    data_t <- us_data[1:t,]
+
+    # Run MCMC to calculate all LPDS
+    res_base <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      LPDS = TRUE, y_test = y_test, x_test = x_test,
+      niter = 60000, nburn = 10000, nthin = 10,
+      hyperprior_param = list(nu_tau = 1, nu_xi = 1))
+
+    res_las_hier <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      LPDS = TRUE, y_test = y_test, x_test = x_test,
+      niter = 60000, nburn = 10000, nthin = 10,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      a_xi = 1, a_tau = 1)
+
+    res_las <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+      LPDS = TRUE, y_test = y_test, x_test = x_test,
+      niter = 60000, nburn = 10000, nthin = 10,
+      learn_a_xi = FALSE, learn_a_tau = FALSE,
+      a_xi = 1, a_tau = 1,
+      learn_kappa2 = FALSE, learn_lambda2 = FALSE,
+      kappa2 = 20, lambda2 = 20)
+
+      res_ng_hier <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+        LPDS = TRUE, y_test = y_test, x_test = x_test,
+        niter = 60000, nburn = 10000, nthin = 10,
+        learn_a_xi = FALSE, learn_a_tau = FALSE,
+        a_xi = 0.1, a_tau = 0.1)
```

```
+
+     res_ng <- shrinkTVP(inf ~ inf_lag + une_lag + tbi_lag, data = data_t,
+       LPDS = TRUE, y_test = y_test, x_test = x_test,
+       niter = 60000, nburn = 10000, nthin = 10,
+       learn_a_xi = FALSE, learn_a_tau = FALSE,
+       a_xi = 0.1, a_tau = 0.1,
+       learn_kappa2 = FALSE, learn_lambda2 = FALSE)
+
+     lpds_res <- c(res_base$LPDS, res_ng_hier$LPDS, res_ng$LPDS,
+       res_las_hier$LPDS, res_las$LPDS)
+
+     rm("res_base", "res_ng", "res_ng_hier", "res_las_hier", "res_las")
+
+     return(lpds_res)
+ }
R> stopCluster(cl)
R>
R> cumu_lpds <- apply(lpds, 1, cumsum)
R> # Plot results
R> par(mar=c(6,4,1,1))
R> colnames(cumu_lpds) <- c("Default", "Hierarchical Normal Gamma",
+    "Non-hierarchical Normal Gamma", "Hierarchical Lasso",
+    "Non-hierarchical Lasso")
R>
R> matplot(cumu_lpds, type = "l", ylab = "Cumulative LPDS", xaxt = "n", xlab = "")
R>
R> # Extraxt labels from time series
R> labs = as.yearmon(time(usmacro.update))[T0:Tmax + 1][c(FALSE, TRUE)]
R>
R> # Create custom axis labels
R> axis(1, at = (1:length(T0:Tmax))[c(FALSE, TRUE)], labels = FALSE)
R> text(x=(1:length(T0:Tmax))[c(FALSE, TRUE)],
+    y=par()$usr[3]-0.05*(par()$usr[4]-par()$usr[3]),
+    labels=labs, srt=45, adj=1, xpd=TRUE)
R>
R> # Add legend
R> legend("topright", colnames(cumu_lpds), col=1:5, lty = 1:5, bty = "n", cex = 0.8)
```

References

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.

Bhadra A, Datta J, Polson N, Willard B (2017). "Lasso Meets Horsheshoe." *https://arxiv.org/abs/1706.10179*.

Bitto A, Frühwirth-Schnatter S (2019). "Achieving Shrinkage in a Time-Varying Parameter Model Framework." *Journal of Econometrics*, **210**, 75–97.

Dangl T, Halling M (2012). "Predictive Regressions with Time-Varying Coefficients." *Journal of Financial Economics*, **106**, 157–181.

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 (2017). "Extending extitR with extitC++: A Brief Introduction to extitRcpp." *PeerJ Preprints*, **5**, e3188v1. ISSN 2167-9843. doi:10.7287/peerj.preprints.3188v1. URL https://doi.org/10.7287/peerj.preprints.3188v1.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R With High-Performance C++ Linear Algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. URL http://dx.doi.org/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. URL http://www.jstatsoft.org/v33/i01/.

Frühwirth-Schnatter S, Wagner H (2010). "Stochastic Model Specification Search for Gaussian and Partially Non-Gaussian State Space Models." **154**, 85–100.

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

Hörmann W, Leydold J (2015). "GIGrvg: Random Variate Generator for the GIG Distribution. R package version 0.4, URL: `http://CRAN.R-project.org/package=GIGrvg`."

Jacquier E, Polson NG, Rossi PE (1994). "Bayesian Analysis of Stochastic Volatility Models." **12**, 371–417.

Kastner G (2016). "Dealing with Stochastic Volatility in Time Series Using the R Package stochvol." *Journal of Statistical Software*, **69**, 1–30.

Kastner G, Frühwirth-Schnatter S (2014). "Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Estimation of Stochastic Volatility Models." *Computational Statistics and Data Analysis*, **76**, 408–423.

Kastner G, Frühwirth-Schnatter S, Lopes HF (2017). "Efficient Bayesian Inference for Multivariate Factor Stochastic Volatility Models." **26**, 905–917.

Knaus P, Bitto-Nemling A, Cadonna A, Frühwirth-Schnatter S (2019). *shrinkTVP: Efficient Bayesian Inference for Time-Varying Parameter Models with Shrinkage*. R package version 1.0.0, URL https://CRAN.R-project.org/package=shrinkTVP.

Krueger 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.

McCausland WJ, Miller S, Pelletier D (2011). "Simulation Smoothing for State Space Models: A Computational Efficiency Analysis." **55**, 199–212.

Microsoft, Weston S (2017). *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.4, URL https://CRAN.R-project.org/package=foreach.

Microsoft, Weston S (2018). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.14, URL https://CRAN.R-project.org/package=doParallel.

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 (2018). *RhpcBLASctl: Control the Number of Threads on 'BLAS'*. R package version 0.18-205, URL https://CRAN.R-project.org/package=RhpcBLASctl.

Park T, Casella G (2008). "The Bayesian Lasso." **103**, 681–686.

Petris G (2010). "An R Package for Dynamic Linear Models." *Journal of Statistical Software*, **36**(12), 1–16. URL http://www.jstatsoft.org/v36/i12/.

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://journal.r-project.org/archive/.

Primiceri G (2005). "Time Varying Structural Vector Autoregressions and Monetary Policy." **72**, 821–852.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Scott SL (2019). *bsts: Bayesian Structural Time Series*. R package version 0.9.1, 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.

Sims CA (2001). "Evolving Post-World War II U.S. Inflation Dynamics: Comment." *NBER Macroeconomics Annual*, **16**, 373–379. ISSN 08893365, 15372642. URL http://www.jstor.org/stable/3585376.

van der Pas S, Scott J, Chakraborty A, Bhattacharya A (2016). *horseshoe: Implementation of the Horseshoe Prior*. R package version 0.1.0, URL https://CRAN.R-project.org/package=horseshoe.

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." **20**, 531–615.

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.

Zeng Y, Breheny P (2017). "The biglasso Package: A Memory- and Computation-Efficient Solver for Lasso Model Fitting with Big Data in R." *ArXiv e-prints*. 1701.05936, URL https://arxiv.org/abs/1701.05936.

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, 4th Floor
1020 Vienna, Austria
E-mail: peter.knaus@wu.ac.at URL: https://www.wu.ac.at/statmath/faculty-staff/faculty/peter-knaus