Modeling Univariate and Multivariate Stochastic Volatility in R with stochvol and factorstochvol

Darjus Hosszejni
WU Vienna University
of Economics and Business

Gregor Kastner
University of Klagenfurt

Abstract

Stochastic volatility (SV) models are nonlinear state-space models that enjoy increasing popularity for fitting and predicting heteroskedastic time series. However, due to the large number of latent quantities, their efficient estimation is non-trivial and software that allows to easily fit SV models to data is rare. We aim to alleviate this issue by presenting novel implementations of five SV models delivered in two R packages. Several unique features are included and documented. As opposed to previous versions, **stochvol** is now capable of handling linear mean models, conditionally heavy tails, and the leverage effect in combination with SV. Moreover, we newly introduce **factorstochvol** which caters for multivariate SV. Both packages offer a user-friendly interface through the conventional R generics and a range of tailor-made methods. Computational efficiency is achieved via interfacing R to C++ and doing the heavy work in the latter. In the paper at hand, we provide a detailed discussion on Bayesian SV estimation and showcase the use of the new software through various examples.

Keywords: Bayesian inference, state-space model, heteroskedasticity, dynamic correlation, dynamic covariance, factor stochastic volatility, Markov chain Monte Carlo (MCMC), leverage effect, asymmetric return distribution, heavy tails, financial time series.

Preface

This vignette corresponds to the article of the same name published in the Journal of Statistical Software (Hosszejni and Kastner 2021a). The version at hand might receive minor updates as time goes by. Further information about citing **stochvol** can be obtained in R by installing the package, e.g., through `install.packages("stochvol")`, and calling `citation("stochvol")`.

1. Introduction

Time dependent variance is an indispensable ingredient of financial and economic time series modeling. Already Markowitz (1952) concerns himself with methods that take into account heteroskedasticity in a better way than a rolling window estimation. By 1982, two fundamentally different approaches had been developed to cater to these needs. On the one hand, Engle (1982) lays the groundwork for a family of time varying volatility models, most notably the

generalized autoregressive conditional heteroskedasticity model (GARCH, Bollerslev 1986). These models feature conditionally deterministic changes in the variance. Taylor (1982), on the other hand, addresses heteroskedasticity in his seminal work with a non-linear latent state space model, later coined the stochastic volatility (SV) model. There, the volatility process evolves in a stochastic manner. Despite some empirical evidence in favor of SV models over their corresponding GARCH counterparts (Jacquier, Polson, and Rossi 1994; Ghysels, Harvey, and Renault 1996; Kim, Shephard, and Chib 1998; Nakajima 2012), SV and its variants enjoy little publicity among practitioners. As Bos (2012) underlines, one reason for this might be the lack of standard software. In response, Kastner (2016) provides a first version of the R (R Core Team 2021) package **stochvol** but fails to feature conditional non-Gaussianity, asymmetry (the so-called leverage effect), and multivariate generalizations.

We address these shortcomings in the manuscript at hand. First, we extend **stochvol** (Hosszejni and Kastner 2021b) with several practically relevant univariate methods. Second, we introduce the new companion package **factorstochvol** (Kastner and Hosszejni 2021) which focuses on the multivariate case. The extended **stochvol** now provides the means for the Bayesian estimation of vanilla SV, heavy-tailed SV, SV with leverage, and heavy-tailed SV with leverage (Harvey and Shephard 1996; Omori, Chib, Shephard, and Nakajima 2007; Nakajima and Omori 2012). Moreover, the package also handles these models naturally when embedded into a linear model or an autoregressive (AR) context. The **factorstochvol** package implements an efficient method for the Bayesian estimation of the factor SV model (Kastner, Frühwirth-Schnatter, and Lopes 2017). Among other features, the package provides several automatic factor identification schemes, hierarchical shrinkage priors (variations of the normal gamma prior, Griffin and Brown 2010), and an array of intuitive visualization methods for the high-dimensional posteriors.

The remainder of this paper is structured as follows. We formally introduce the univariate and the multivariate models in Sections 2 and 3, respectively, including a discussion about prior distributions and a brief overview of the estimation methods. In Section 4, we unveil the new samplers of the **stochvol** package through three example models. We describe the **factorstochvol** package in Section 5, and then we conclude.

2. Univariate SV models

We begin by introducing the vanilla SV model with linear regressors, henceforth simply called the SV model. This is a minor but important extension of the SV model without regressors. We also settle the notation and establish a baseline model that we generalize and reuse throughout the manuscript. Consequently, we proceed with three generalized models: the SV model with Student's t errors (SVt), the SV model with leverage (SVl), and their combination, the SV model with Student's t errors and leverage (SVtl). Finally, we close the section after discussing prior distributions and Markov chain Monte Carlo (MCMC) sampling.

2.1. Model specifications

The key feature of the SV model is its stochastic and time-varying specification of the variance evolution. In particular, the log-variance is assumed to follow an AR(1) process. This feature unites the following models.

Vanilla SV with linear regressors

Let $\boldsymbol{y} = (y_1, \dots, y_n)^\top$ denote a vector of observations. The SV model assumes the following structure for $\boldsymbol{y}$,

$$\begin{aligned} y_t &= \boldsymbol{x}_t \boldsymbol{\beta} + \exp(h_t/2)\varepsilon_t, \\ h_{t+1} &= \mu + \varphi(h_t - \mu) + \sigma\eta_t, \\ \varepsilon_t &\sim \mathcal{N}(0,1), \\ \eta_t &\sim \mathcal{N}(0,1), \end{aligned} \tag{1}$$

where $\mathcal{N}(b, B)$ denotes the normal distribution with mean $b \in \mathbb{R}$ and variance $B \in \mathbb{R}^+$, and ε_t and η_t are independent. The log-variance process $\boldsymbol{h} = (h_1, \dots, h_n)^\top$ is initialized by $h_0 \sim \mathcal{N}(\mu, \sigma^2/(1-\varphi^2))$. $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_n^\top)^\top$ is an $n \times K$ matrix containing in its tth row the vector of K regressors at time t. The K regression coefficients are collected in $\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)^\top$. We refer to $\boldsymbol{\vartheta} = (\mu, \varphi, \sigma)$ as the SV parameters: μ is the level, φ is the persistence, and σ (also called *volvol*) is the standard deviation of the log-variance.

SV with Student's t errors

Several authors have suggested to use non-normal conditional residual distributions for stochastic volatility modeling. Examples include the Student's t distribution (Harvey, Ruiz, and Shephard 1994), the extended generalized inverse Gaussian (Silva, Lopes, and Migon 2006), (semi-)parametric residuals (Jensen and Maheu 2010; Delatola and Griffin 2011), or the generalized hyperbolic skew Student's t distribution (Nakajima and Omori 2012). We implement Student's t errors for the observation equation in **stochvol**. Formally,

$$\begin{aligned} y_t &= \boldsymbol{x}_t \boldsymbol{\beta} + \exp(h_t/2)\varepsilon_t, \\ h_{t+1} &= \mu + \varphi(h_t - \mu) + \sigma\eta_t, \\ \varepsilon_t &\sim t_\nu(0,1), \\ \eta_t &\sim \mathcal{N}(0,1), \end{aligned} \tag{2}$$

where ε_t and η_t are independent. $t_\nu(a, b)$ is the Student's t distribution with ν degrees of freedom, mean a, and variance b. The single difference between Equation 1 and Equation 2 is that here the observations are conditionally t distributed. Hence, Equation 2 generalizes Equation 1 through the new parameter ν as the Student's t distribution converges in law to the standard normal distribution when ν goes to infinity.

SV with leverage

Propositions for asymmetric innovations include non-parametric distributions (Jensen and Maheu 2014), skewed distributions (Nakajima and Omori 2012), and distributions featuring correlation with the variance process, also called the leverage effect (Harvey and Shephard 1996; Jacquier, Polson, and Rossi 2004). We implement the leverage effect in the **stochvol** package. Formally,

$$\begin{aligned} y_t &= \boldsymbol{x}_t \boldsymbol{\beta} + \exp(h_t/2)\varepsilon_t, \\ h_{t+1} &= \mu + \varphi(h_t - \mu) + \sigma\eta_t, \\ \varepsilon_t &\sim \mathcal{N}(0,1), \\ \eta_t &\sim \mathcal{N}(0,1), \end{aligned} \tag{3}$$

where the correlation matrix of (ε_t, η_t) is

$$\boldsymbol{\Sigma}^{\rho} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}. \tag{4}$$

The vector $\boldsymbol{\zeta} = (\mu, \varphi, \sigma, \rho)^{\top}$ collects the SV parameters. The new parameter compared to Equation 1 is a correlation term ρ which relates the residuals of the observations to the innovations of the variance process. Equation 1 is therefore a special case of Equation 3 with a pre-fixed $\rho = 0$.

SV with Student's t errors and leverage

Some authors have proposed the combination of t errors with the leverage effect (Jacquier *et al.* 2004; Omori *et al.* 2007; Nakajima and Omori 2009). We implement the common generalization of Equation 2 and Equation 3. Formally,

$$\begin{aligned} y_t &= \boldsymbol{x}_t\boldsymbol{\beta} + \exp(h_t/2)\varepsilon_t, \\ h_{t+1} &= \mu + \varphi(h_t - \mu) + \sigma\eta_t, \\ \varepsilon_t &\sim t_{\nu}(0, 1), \\ \eta_t &\sim \mathcal{N}(0, 1), \end{aligned} \tag{5}$$

where the correlation matrix of (ε_t, η_t) is $\boldsymbol{\Sigma}^{\rho}$ as in Equation 4.

2.2. Prior distributions

We a priori assume $\boldsymbol{\beta} \sim \mathcal{N}_K(\boldsymbol{b}_{\boldsymbol{\beta}}, \boldsymbol{B}_{\boldsymbol{\beta}})$, where $\mathcal{N}_l(\boldsymbol{b}, \boldsymbol{B})$ is the l-dimensional normal distribution with mean vector $\boldsymbol{b}$ and variance-covariance matrix $\boldsymbol{B}$. For small values in the diagonal of $\boldsymbol{B}_{\boldsymbol{\beta}}$, this prior enforces shrinkage towards $\boldsymbol{b}_{\boldsymbol{\beta}}$; for large values in the diagonal, the prior turns rather uninformative. By setting $\boldsymbol{b}_{\boldsymbol{\beta}}$ to the zero vector and $\boldsymbol{B}_{\boldsymbol{\beta}}$ to a scaled identity matrix, the prior distribution becomes the Bayesian analogue to ridge regression (see, e.g., Park and Casella 2008, for a discussion of this and other shrinkage priors).

The level $\mu \in \mathbb{R}$ is unrestricted, hence we can apply the common $\mu \sim \mathcal{N}(b_{\mu}, B_{\mu})$ prior. Depending on the application, a fairly uninformative distribution is the usual choice, e.g., setting $b_{\mu} = 0$ and $B_{\mu} \geq 100$ for daily asset log returns. In our experience, the exact values of the prior mean and prior variance of μ do not strongly affect the estimation results unless B_{μ} is small.

To achieve stationarity in the variance process, a restricted persistence $\varphi \in (-1, 1)$ is needed. To this end, we assume $(\varphi + 1)/2 \sim \mathcal{B}(a_{\varphi}, b_{\varphi})$, where $\mathcal{B}(a_{\varphi}, b_{\varphi})$ is the beta distribution with shape parameters a_{φ} and b_{φ}. The selection of the shape parameters may be relatively influential with many data sets. In financial applications with daily asset log returns, the variance tends to be highly persistent, i.e., $\varphi \approx 1$. Such domain knowledge can be used as prior information by allocating more probability to positive high values of φ, e.g., by setting $a_{\varphi} \gtrsim 5$ and $b_{\varphi} \approx 1.5$. As an alternative, when stationarity is not assumed, the untruncated prior $\varphi \sim \mathcal{N}(b_{\varphi}, B_{\varphi})$ can also be applied.

The volvol is positive but we would like allow σ to approach 0 as closely as needed – that allows us to be less informative and to improve the estimates. Following Frühwirth-Schnatter and Wagner (2010) and Kastner and Frühwirth-Schnatter (2014), we advocate

$\sigma \sim |\mathcal{N}(0, B_\sigma)|$ instead, where $|\mathcal{N}(0, B_\sigma)|$ denotes the half normal distribution. It corresponds to $\sigma^2 \sim \mathcal{G}(1/2, 1/(2B_\sigma))$, where $\mathcal{G}(a, b)$ is the gamma distribution with shape parameter a and rate parameter b. As an alternative, the commonly applied and convenient conjugate gamma prior on σ^{-2} can be assumed. However, it bounds σ away from 0 and it is therefore in our view an unsatisfactory choice.

As a last step in fully specifying the vanilla SV model in Equation 1, the variance process is initialized a priori with its stationary distribution, i.e., $h_0 \sim \mathcal{N}(\mu, \sigma^2/(1-\varphi^2))$. This consistently extends our prior assumptions about $\boldsymbol{h}$ following a stationary AR(1) process. As an alternative, when stationarity is not assumed, $h_0 \sim \mathcal{N}(\mu, B_h)$ can be applied with a constant variance B_h.

The SV models with Student's t errors additionally require the prior specification of the degrees of freedom parameter ν. To ascertain interpretability of the scaling $\exp(h_t/2)$, we ensure finite second moments of $\boldsymbol{y}$ by enforcing $\nu > 2$. As a reviewer recommended, we follow (Geweke 1993) and equip ν with an exponential prior $\nu - 2 \sim \mathcal{E}(\lambda_\nu)$, where λ_ν is the rate of the exponential distribution.

Finally, in the case of the SV models with leverage, we employ the translated and scaled beta distribution for $\rho \in (-1, 1)$ as in Omori *et al.* (2007), i.e., $(\rho + 1)/2 \sim \mathcal{B}(a_\rho, b_\rho)$. We find that the posterior estimates of ρ can be sensitive to its prior distribution, thus, some care is needed when setting the hyperparameters in practice. In our experience, slightly informative choices such as $a_\rho = b_\rho \approx 4$ work well in financial applications.

2.3. Estimation

All methods implemented in **stochvol** and **factorstochvol** rely on the Bayesian paradigm.[1] Bayesian analysis aims to estimate model parameters through Bayesian updating. By using probability distributions to represent information, Bayes' theorem can be employed to update the prior information to the posterior information by incorporating the observations. This approach has the advantage of providing full uncertainty quantification in a probabilistic framework without relying on asymptotic results; moreover, so-called *shrinkage priors* can be used to regularize the posterior and guard against overfitting. For an introductory textbook on Bayesian statistics, see, for instance, McElreath (2015).

When the posterior distribution is not available analytically, one customarily resorts to approximations such as perfect simulation (Huber 2015), approximate Bayesian computation (Sisson, Fan, and Beaumont 2018), adaptive Monte Carlo methods (Roberts and Rosenthal 2007), or MCMC methods. When computationally feasible, MCMC is a valuable tool that provides draws from the posterior distribution in question. That way, MCMC approximates the posterior distribution similarly to a histogram approximating a density. For a more in-depth introduction on MCMC methods, see, for instance, Brooks, Gelman, Jones, and Meng (2011).

The estimation algorithm of SV, SVt, SVl, and SVtl all resemble the original methodology developed in Kastner and Frühwirth-Schnatter (2014) for the vanilla SV model. Namely, to draw from the posterior distribution of $\boldsymbol{h}$ efficiently, the MCMC sampler resorts to approximate mixture representations of Equations 1, 2, 3 and 5 similar to the ones in Kim *et al.*

[1] At this point we would also like to point out works aiming at estimating stochastic volatility and related models within the frequentist framework, see, e.g., Abanto-Valle, Langrock, Chen, and Cardoso (2017); Creal (2017), and in particular the recent **stochvolTMB** package (Wahl 2020).

(1998) and Omori *et al.* (2007). Doing so yields a conditionally Gaussian state space model for which efficient sampling methods are available (Frühwirth-Schnatter 1994; Carter and Kohn 1994). Following Rue (2001) and McCausland, Miller, and Pelletier (2011), we draw the full vector $\boldsymbol{h}$ "all without a loop" (AWOL).

When Student's t errors with unknown degrees of freedom are used, we handle the added complication through the well-known representation of the t distribution as a scale mixture of Gaussians. This requires additional Gibbs and independence Metropolis-Hastings steps documented in Kastner (2015). Furthermore, we deal with the increased complexity in the posterior space of the leverage case by repeated ancillarity-sufficiency interweaving strategies (ASIS, Yu and Meng 2011) steps in the sampling scheme, see Hosszejni and Kastner (2019) for details.

To verify the correctness of the implementation, unit tests are included in the package which can be run by `devtools::test()` (Wickham, Hester, and Chang 2020). In particular, a variant of Geweke's test (Geweke 2004) is part of the test suite. In this test, we exploit that the sampling distribution of the model parameters during the Geweke test is identical to their preset prior distribution. Therefore, the cumulative distribution function maps the sample to a uniform distribution, which in turn is mapped to a normal distribution using the normal distribution's quantile function. If the user chooses to execute the automated unit tests in **stochvol**, the system evaluates the thinned and transformed sample using the `shapiro.test()` function, where the thinning of the sample is done to approximate independent sampling.

For maximal computational effectiveness, all sampling algorithms are implemented in the compiled language C++ (ISO/IEC 2017) with the help of the R package **Rcpp** (Eddelbuettel and François 2011). Matrix computations make use of the efficient C++ template library **Armadillo** (Sanderson and Curtin 2016) through the R package **RcppArmadillo** (Eddelbuettel and Sanderson 2014).[2] After sampling, the resulting R objects make use of plotting and summary functions of the R package **coda** (Plummer, Best, Cowles, and Vines 2006).

3. Multivariate SV models

A key difficulty accompanying dynamic covariance estimation is the relatively high number of unknowns compared to the number of observations. More precisely, letting m denote the cross-sectional dimension, the corresponding covariance matrix $\boldsymbol{\Sigma}_t$ contains $m(m+1)/2$ degrees of freedom, a quadratic term in m. Table 1 illustrates the "curse of dimensionality" for various values of m. One way to break this curse is to use latent factors and thereby achieve a sparse representation of $\boldsymbol{\Sigma}_t$.

3.1. The factor SV model

Latent factor models embody the idea that even high dimensional systems are driven by only a few sources of randomness. These few sources of randomness control a few factors, which in turn account for the interactions between the observations. Moreover, latent factor models provide an efficient tool for dynamic covariance matrix estimation. They allow for a reduction in the number of unknowns. A conventional latent factor model with r factors implies the

[2]For explicit run time discussions please see Kastner and Frühwirth-Schnatter (2014) and Hosszejni and Kastner (2019). For the possibility to use multi-core computing within a single MCMC chain and potential speed gains when doing so, please see Kastner (2019).

m	free elements of $\mathbf{\Sigma}_t$	free elements of $\mathbf{\Sigma}_t$ per data point
1	1	1
10	55	5.5
100	5050	50.5
1000	500500	500.5

Table 1: Absolute and relative numbers of free elements of the time-varying covariance matrix $\mathbf{\Sigma}_t$ for different numbers of component series m.

m	free elements of $\mathbf{\Sigma}_t$	free elements of $\mathbf{\Sigma}_t$ per data point
10	44	4.4
100	494	4.94
1000	4994	4.994

Table 2: Absolute and relative numbers of free elements of the time-varying covariance matrix $\mathbf{\Sigma}_t$ in a factor model for different numbers of component series m and number of factors $r = 4$.

decomposition

$$\mathbf{\Sigma}_t = \check{\mathbf{\Sigma}}_t + \bar{\mathbf{\Sigma}}_t, \tag{6}$$

where $\operatorname{rank}(\check{\mathbf{\Sigma}}_t) = r < m$, and $\bar{\mathbf{\Sigma}}_t$ is the diagonal matrix containing the variances of the idiosyncratic errors. The rank assumption on the symmetric $\check{\mathbf{\Sigma}}_t$ gives rise to the factorization $\check{\mathbf{\Sigma}}_t = \Psi\Psi^\top$, where $\Psi \in \mathbb{R}^{m\times r}$ contains $mr - r(r-1)/2$ free elements (see, e.g., the pivoted Cholesky algorithm in Higham 1990). Hence, $m(r+1) - r(r-1)/2$ free elements remain in $\mathbf{\Sigma}_t$, now only linear in m. Table 2 illustrates the "broken curse of dimensionality" for various values of m and $r = 4$.

In the following, we describe the factor SV model employed in the **factorstochvol** package. We model the observations $\boldsymbol{y}_t = (y_{t1}, \ldots, y_{tm})^\top$ as follows.

$$\begin{aligned} \boldsymbol{y}_t \mid \boldsymbol{\beta}, \mathbf{\Lambda}, \boldsymbol{f}_t, \bar{\mathbf{\Sigma}}_t &\sim \mathcal{N}_m(\boldsymbol{\beta} + \mathbf{\Lambda}\boldsymbol{f}_t, \bar{\mathbf{\Sigma}}_t), \\ \boldsymbol{f}_t \mid \tilde{\mathbf{\Sigma}}_t &\sim \mathcal{N}_r(\mathbf{0}, \tilde{\mathbf{\Sigma}}_t), \end{aligned} \tag{7}$$

where $\boldsymbol{f}_t = (f_{t1}, \ldots, f_{tr})^\top$ is the vector of factors, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)^\top$ is an observation-specific mean, and $\mathbf{\Lambda} \in \mathbb{R}^{m\times r}$ is a tall matrix holding the factor loadings. The covariance matrices $\bar{\mathbf{\Sigma}}_t$ and $\tilde{\mathbf{\Sigma}}_t$ are both diagonal representing independent vanilla SV processes.

$$\begin{aligned} \bar{\mathbf{\Sigma}}_t &= \operatorname{diag}(\exp(\bar{h}_{t1}), \ldots, \exp(\bar{h}_{tm})), \\ \tilde{\mathbf{\Sigma}}_t &= \operatorname{diag}(\exp(\tilde{h}_{t1}, \ldots, \exp(\tilde{h}_{tr}))), \\ \bar{h}_{ti} &\sim \mathcal{N}(\bar{\mu}_i + \bar{\varphi}_i(\bar{h}_{t-1,i} - \bar{\mu}_i), \bar{\sigma}_i^2), \quad i = 1, \ldots, m, \\ \tilde{h}_{tj} &\sim \mathcal{N}(\tilde{\mu}_j + \tilde{\varphi}_j(\tilde{h}_{t-1,j} - \tilde{\mu}_j), \tilde{\sigma}_j^2), \quad j = 1, \ldots, r. \end{aligned} \tag{8}$$

For a more theoretical treatment of factor SV from a Bayesian point of view, the reader is referred to, e.g., Pitt and Shephard (1999), Aguilar and West (2000), Chib, Nardari, and Shephard (2006), and Han (2006).

Based on Equation 7, we can reformulate Equation 6 as

$$\mathbf{\Sigma}_t = \mathbf{\Lambda}\tilde{\mathbf{\Sigma}}_t\mathbf{\Lambda}^\top + \bar{\mathbf{\Sigma}}_t, \tag{9}$$

from which several identification issues are apparent: the order, the sign, and the scale of the factors is unidentified. More specifically, for any generalized permutation matrix[3] $\boldsymbol{P}$ of size $r \times r$, we find another valid decomposition $\boldsymbol{\Sigma}_t = \boldsymbol{\Lambda}'\tilde{\boldsymbol{\Sigma}}_t'(\boldsymbol{\Lambda}')^\top + \bar{\boldsymbol{\Sigma}}_t$, where $\boldsymbol{\Lambda}' = \boldsymbol{\Lambda}\boldsymbol{P}^{-1}$ and $\tilde{\boldsymbol{\Sigma}}_t' = \boldsymbol{P}\tilde{\boldsymbol{\Sigma}}_t\boldsymbol{P}^\top$. We resolve the ambiguity in the scale of the factors by fixing the level of their log-variance to zero, i.e., $\tilde{\mu}_j = 0$ for $j = 1, \ldots, r$. Sign and order identification can be enforced through restrictions on the factor loadings matrix $\boldsymbol{\Lambda}$. Several options are available in **factorstochvol** for restricting $\boldsymbol{\Lambda}$, for details see Section 5.2.

3.2. Prior distributions

Priors need to be specified for the mean, the latent log-variance processes, and for the factor loadings matrix $\boldsymbol{\Lambda}$. We choose $\beta_j \sim \mathcal{N}(b_\beta, B_\beta)$, independently for $j = 1, \ldots, m$. For small values of B_β, this shrinks β_j toward b_β; for large values of B_β, the prior is fairly uninformative.

The log-variance processes have the same prior specification as in the univariate case in Section 2.2. For $\boldsymbol{\Lambda}$, three types of priors are currently implemented in **factorstochvol**. All three can be written in the form $\Lambda_{ij} \sim \mathcal{N}(0, \tau_{ij}^2)$ independently for each applicable $i \in \{1, \ldots m\}$ and $j \in \{1, \ldots, r\}$. First, one can fix all the τ_{ij}^2s – not necessarily to the same value – a priori. This results in a normal prior for each element of the loadings matrix.

The second type is a hierarchical prior which has been developed to induce more flexible and potentially stronger shrinkage,

$$\Lambda_{ij} \mid \tau_{ij}^2 \sim \mathcal{N}(0, \tau_{ij}^2), \quad \tau_{ij}^2 \mid \lambda_i^2 \sim \mathcal{G}(a, a\lambda_i^2/2). \tag{10}$$

This distribution is termed normal gamma prior by Griffin and Brown (2010) and implies a conditional variance $\mathbb{V}(\Lambda_{ij} \mid \lambda_i^2)$ of $2/\lambda_i^2$ and an unconditional excess kurtosis of $3/a$. The value of a is treated as a structural parameter to be fixed by the user, where choosing a small ($\lesssim 1$) enforces strong shrinkage towards zero, while choosing a large ($\gtrsim 1$) imposes little shrinkage. The case $a = 1$ is a special case termed the Bayesian Lasso prior (Park and Casella 2008). The parameter λ_i^2 is estimated from the data with $\lambda_i^2 \sim \mathcal{G}(c, d)$.

The third type is a slight modification of the second. Because variances in each row of the factor loadings matrix $\boldsymbol{\Lambda}$ can be seen as "random effects" from the same underlying distribution, the prior in Equation 10 induces row-wise shrinkage with element-wise adaption. Analogously, one could also consider column-wise shrinkage with element-wise adaption, i.e.,

$$\Lambda_{ij} \mid \tau_{ij}^2 \sim \mathcal{N}(0, \tau_{ij}^2), \quad \tau_{ij}^2 \mid \lambda_j^2 \sim \mathcal{G}(a, a\lambda_j^2/2), \tag{11}$$

with the corresponding prior $\lambda_j^2 \sim \mathcal{G}(c, d)$.

3.3. Estimation

Bayesian estimation in the factor SV model builds on the univariate vanilla SV implementations in **stochvol** and features several levels of efficiency boosting. To alleviate the problem of potentially slow convergence in high dimensions, it is carried out via a sampler that utilizes several variants of ASIS. The sampling details implemented in **factorstochvol** are described in

[3] A generalized permutation matrix has the zero–non-zero pattern of a permutation matrix, but it is allowed to have any non-zero values instead of just ones. Hence, a generalized permutation matrix not only permutes but also scales and switches the sign of its multiplier.

Kastner *et al.* (2017, using Gaussian priors for the factor loadings) as well as Kastner (2019, using hierarchical shrinkage priors for the factor loadings).

Similarly to **stochvol** and in an attempt to make computation time bearable even in higher dimensions, **factorstochvol**'s main sampler is written in C++. It uses the R package **Rcpp** to ease communication between R and C++. The univariate SV parts are borrowed from **stochvol** and interfaced through its C/C++-level updating function `update_fast_sv()`. In doing so, moving between interpreted R code and compiled C++ code at each MCMC iteration is avoided.

4. The stochvol package

The **stochvol** package provides means for fitting univariate SV, SVt, SVl, and SVtl models via its sampling routines `svsample()`, `svtsample()`, `svlsample()`, and `svtlsample()`, respectively. In the following, we describe a recommended workflow with **stochvol**. First, we discuss estimation, visualization, and prediction using default settings. Then, we show how to adapt the values of the prior hyperparameters and how to configure the sampling mechanism.

4.1. Preparing the data and running the MCMC sampler

We estimate three models that exemplify the features and the user interface of **stochvol**. Using the `exrates` data found in the package, we model the EURCHF exchange rate (the price of 1 euro in Swiss franc) in the period between March 1, 2011 and March 1, 2012 (260 data points) in three different ways.[4]

AR(1) model with SV residuals

The first example is an AR(1) model with SV residuals, i.e., Equation 1 turns into

$$y_t \mid y_{t-1}, \beta_0, \beta_1, h_t \sim \mathcal{N}(\beta_0 + \beta_1 y_{t-1}, \exp(h_t)),$$
$$h_{t+1} \mid \boldsymbol{\vartheta}, h_t \sim \mathcal{N}(\mu + \varphi(h_t - \mu), \sigma^2).$$

Using this model, we test whether the exchange rate follows a random walk with SV. In this case, we expect the posteriors of β_0 and β_1 to concentrate around 0 and 1, respectively.

In order to estimate this AR(1)-SV model, we need to prepare the input $\boldsymbol{y}$ as a `numeric` sequence of length n and pass it as the first input argument to `svsample()` as follows:

```
R> set.seed(1)
R> library("stochvol")
R> data("exrates")
R> ind <- which(exrates$date >= as.Date("2011-03-01") &
+    exrates$date <= as.Date("2012-03-01"))
R> CHF_price <- exrates$CHF[ind]
R> res_sv <- svsample(CHF_price, designmatrix = "ar1")
```

[4]In the original article, the period between March 1, 2008 and March 1, 2012 (1028 data points) is used. Here, to reduce compilation time and document size for CRAN, this shorter period is employed.

We set `designmatrix = "ar1"` to use the AR(1) specification. More generally, `designmatrix` can take `character` values of the form `"ar0"` for a constant mean model, or `"ar1"`, `"ar2"`, etc., for AR(1), AR(2), and so on.

Constant mean model with SVt residuals

The second example is a constant mean model with SVt residuals, i.e., Equation 2 becomes

$$y_t \mid \beta_0, h_t, \nu \sim t_\nu(\beta_0, \exp(h_t/2)),$$
$$h_{t+1} \mid \boldsymbol{\vartheta}, h_t \sim \mathcal{N}(\mu + \varphi(h_t - \mu), \sigma^2).$$

If the returns are heavy-tailed, most of the posterior mass of ν concentrates on low values, e.g., smaller than 20. Otherwise, there is little evidence for high kurtosis.

We compute the log returns by applying `logret()` on the previously calculated `CHF_price`. Then, to estimate the constant mean model with heavy tailed SV residuals, we pass the vector of log returns to `svtsample()` with `designmatrix` set to `"ar0"`.

```
R> set.seed(2)
R> CHF_logret <- 100 * logret(CHF_price)
```

```
R> set.seed(2)
R> CHF_logret <- 100 * logret(CHF_price)
R> res_svt <- svtsample(CHF_logret, designmatrix = "ar0")
```

Multiple regression with SVl residuals

The third example is a multiple regression model with an intercept, two regressors, and SVl residuals; that is, Equation 3 turns into

$$\begin{pmatrix} y_t \\ h_{t+1} \end{pmatrix} \,\Bigg|\, h_t, \boldsymbol{\zeta}, \begin{pmatrix} x_{t1} \\ x_{t2} \end{pmatrix}, \beta_0, \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \sim \mathcal{N}_2 \left(\begin{pmatrix} \beta_0 + \beta_1 x_{t1} + \beta_2 x_{t2} \\ \mu + \varphi(h_t - \mu) \end{pmatrix}, \boldsymbol{\Sigma}^\rho \right),$$
$$\boldsymbol{\Sigma}^\rho = \begin{pmatrix} \exp(h_t) & \rho\sigma \exp(h_t/2) \\ \rho\sigma \exp(h_t/2) & \sigma^2 \end{pmatrix}.$$

For illustration, we regress EURCHF log returns onto the contemporaneous log returns on EURUSD and EURJPY, the value of 1 euro per US dollar and Japanese yen, respectively.

To estimate a multiple regression model using **stochvol**, we need to prepare a `numeric` matrix $\boldsymbol{X}$ of dimension $n \times K$, where rows correspond to time points and columns to covariates. We create an intercept as the first column of $\boldsymbol{X}$, and we set the second and the third columns to the EURUSD log returns and the EURJPY log returns, respectively; finally, we use the columns of $\boldsymbol{X}$ as covariates in the multiple regression.[5]

```
R> set.seed(3)
R> X <- cbind(constant = 1, 100 * logret(exrates$USD[ind]),
+    100 * logret(exrates$JPY[ind]))
R> res_svl <- svlsample(CHF_logret, designmatrix = X,
+    thin = 20)
```

[5]Thinning is applied in order to reduce the size of the PDF document on CRAN.

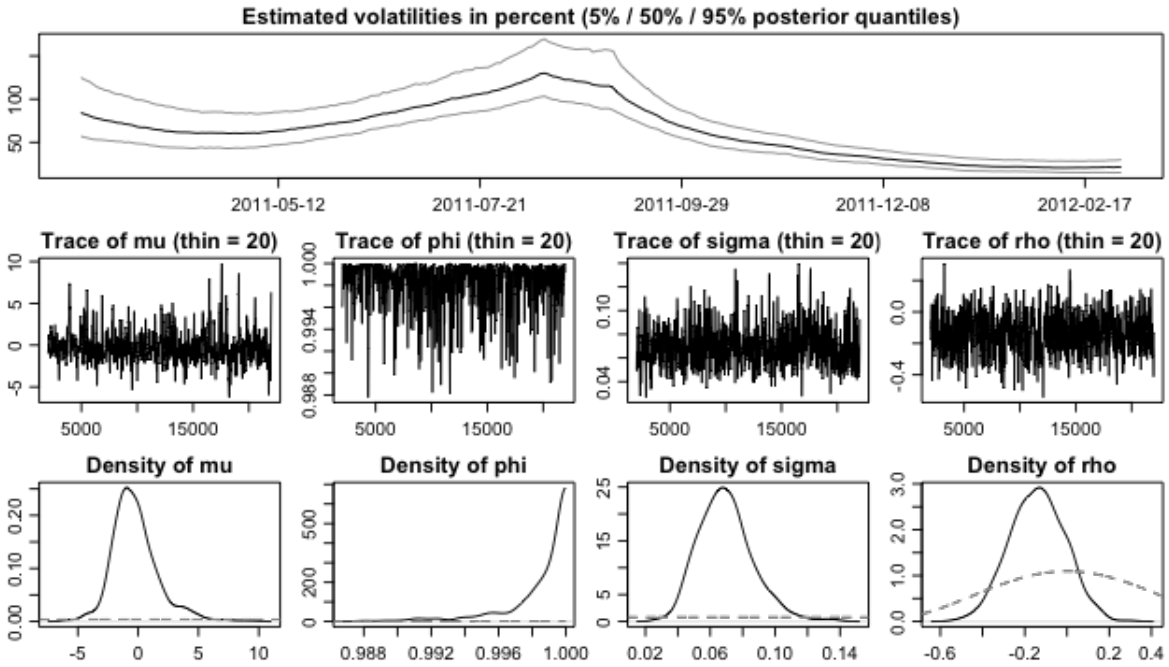

Figure 1: The default plot of an estimated model. The top row shows a summary of the posterior of the daily volatility (in percent) $100 \exp(\boldsymbol{h}/2)$ through its median (black) and 5% and 95% quantiles (gray). The remaining panels summarize the Markov chains of the parameters μ, φ, σ, and ρ. In particular, the middle row presents trace plots and the bottom row shows prior (gray, dashed) and posterior (black, solid) densities.

4.2. Visualizing the results

Often, the joint posterior distribution of model parameters and latent quantities mark the goal of a Bayesian analysis. To inspect it, one can look at summary statistics and various types of visualizations of marginal posterior distributions. Also, it is recommended to examine the Markov chain for possible convergence issues – this happens usually by investigating trace plots of posterior quantities. For this reason, inspired by the **coda** package, **stochvol** provides its own instances of the R generic functions `plot()` and `summary()`. In order to introduce the tools that **stochvol** provides for analyzing MCMC output, we briefly examine the results of the third example (multiple regression with SVl errors) in the remaining part of the section.

First, we plot the output of the estimation.

```
R> plot(res_svl, showobs = FALSE, dates = exrates$date[ind[-1]])
```

The result is shown in Figure 1. We see in the first row a summary of the posterior density of the volatility. Apart from its median, we also receive a quantification of the uncertainty through the 5% and the 95% quantiles at each time point. In the second row, we can follow the evolution of the Markov chain of the SV parameters. In this example, they are μ, φ, σ, and ρ. Lastly, we see prior and posterior density plots of the parameters in the third row in gray and black, respectively. They show high persistence and significant leverage.

Next, we observe the AR coefficients.

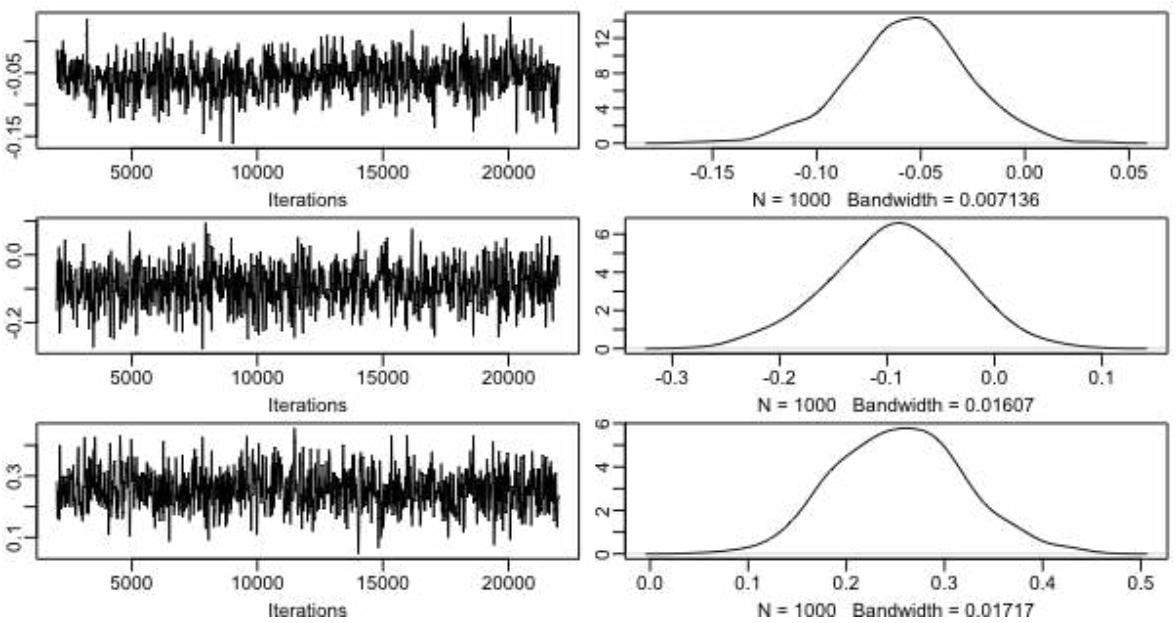

Figure 2: Trace plots and estimated kernel densities of posterior draws from $p(\boldsymbol{\beta} \mid \boldsymbol{y})$.

```
R> for (i in seq_len(3)) {
+    coda::traceplot(svbeta(res_svl)[, i])
+    coda::densplot(svbeta(res_svl)[, i], show.obs = FALSE)
+  }
```

The result is shown in Figure 2. On the left hand side, we do not spot any signs of convergence or mixing problems in the trace plots. On the right hand side, we see that none of the posterior densities of β_0, β_1, and β_2 concentrate around 0, hence the covariates seem to have an impact on the dependent variable.

As the final step, we print a numeric summary of the estimation results.

```
R> summary(res_svl, showlatent = FALSE)


Summary of 'svdraws' object

Prior distributions:
mu        ~ Normal(mean = 0, sd = 100)
(phi+1)/2 ~ Beta(a = 5, b = 1.5)
sigma^2   ~ Gamma(shape = 0.5, rate = 0.5)
nu        ~ Infinity
rho       ~ Beta(a = 4, b = 4)
beta      ~ MultivariateNormal(...)

Stored 1000 MCMC draws after a burn-in of 2000.
Thinning: 20.
```

```
Posterior draws of SV parameters (thinning = 20):
             mean     sd     5%     50%    95% ESS
mu        -0.2385 1.9430 -2.714 -0.4573 3.5228 294
phi        0.9983 0.0021  0.994  0.9991 1.0000 727
sigma      0.0695 0.0167  0.045  0.0682 0.0996 635
rho       -0.1399 0.1296 -0.352 -0.1389 0.0700 624
exp(mu/2)  1.8965 6.5278  0.257  0.7956 5.8208 294
sigma^2    0.0051 0.0025  0.002  0.0046 0.0099 635

Posterior draws of regression coefficients (thinning = 20):
         mean    sd    5%    50%     95%  ESS
beta_0 -0.056 0.029 -0.11 -0.055 -0.0096 1000
beta_1 -0.091 0.060 -0.20 -0.089  0.0036 1065
beta_2  0.255 0.064  0.15  0.254  0.3635  960
```

For brevity, we set `showlatent = FALSE` in order not to print all the 1027 latent states. The output shows the length of the burn-in and the number of draws, the prior specification of the parameters, and a concise summary of the marginal posterior distributions of the parameters μ, φ, σ, and ρ, and additionally of the level of the volatility $\exp(\mu/2)$ and of σ^2, and of the vector of regression coefficients $\boldsymbol{\beta}$. This posterior summary is a table consisting of columns for the posterior mean and standard deviation, the 5%, 50%, and 95% quantiles. The user can influence the shown quantiles by passing a sequence of values between 0 and 1 to `svsample()`, `svtsample()`, `svlsample()`, or `svtlsample()` via the argument `quantiles`.

The last column in the table depicts the so-called effective sample size (ESS), a measure of the quality of a converged MCMC chain. Formally, ESS of a Markov chain C is defined through $M/(1 + 2\sum_{s=1}^{\infty} \rho^{\text{eff}}(s))$, where M is the length of C and $\rho^{\text{eff}}(s)$ denotes the autocorrelation function for lag s among the elements of C. In principle, ESS is the sample size of a serially uncorrelated chain bearing the same Monte Carlo error as our (marginal) chain. Intuitively speaking, this means that ESS is the number of independent and identically distributed draws that were acquired and gives a sense of how well our chain has explored the posterior space. Higher values of ESS indicate better mixing.

4.3. Prediction with stochvol

We employ our estimated model to predict log returns for the remaining days in the data set. To do so, we first prepare the covariates for the next 4 days and pass them via the argument `newdata` of the generic `predict()` function along with the estimation output. Note that we need 5 days of price data to obtain 4 returns.[6]

```
R> set.seed(4)
R> pred_ind <- seq(tail(ind, 1), length.out = 5)
R> pred_X <- cbind(constant = 1, 100 * logret(exrates$USD[pred_ind]),
+    100 * logret(exrates$JPY[pred_ind]))
R> pred_svl <- predict(res_svl, 4, newdata = pred_X)
```

As we have access to the entire distribution of future log returns, we can quantify the uncertainty around our predictions through quantiles. In the following code snippet, we visualize

[6] In the original document, the next 25 days are predicted; here, we restrict ourselves to the next five days.

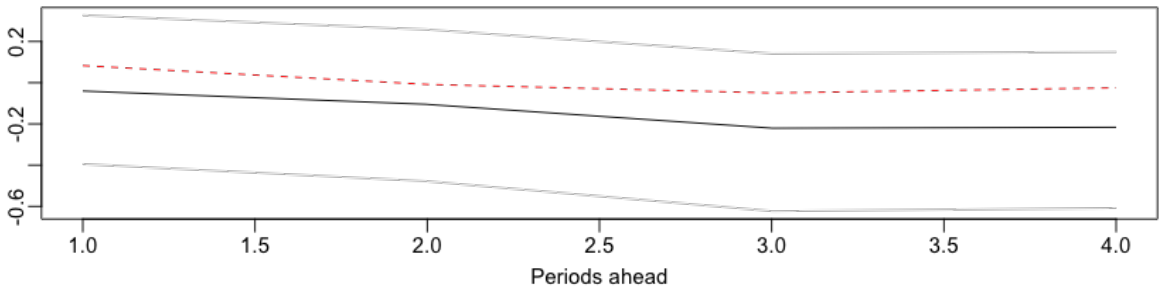

Figure 3: Multi-step ahead predictive distributions (solid, gray and black) and observations (dashed, red).

the k-step-ahead predictive distributions for $k = 1, \ldots, 24$, along with the truly observed values. The result is in Figure 3.

```
R> obs_CHF <- 100 * logret(exrates$CHF[pred_ind])
R> qs <- t(apply(predy(pred_svl), 2, quantile, c(0.05, 0.5, 0.95)))
R> ts.plot(cbind(qs, obs_CHF), xlab = "Periods ahead", lty = c(rep(1, 3), 2),
+    col = c("gray80", "black", "gray80", "red"))
```

4.4. Rolling window estimation

Inspired by `ugarchroll()` in the R package **rugarch** (Ghalanos 2020), we introduce the suite of wrapper functions `svsample_roll()`, `svtsample_roll()`, `svlsample_roll()`, and `svtlsample_roll()`, built around their corresponding routines `svsample()`, `svtsample()`, `svlsample()`, and, respectively, `svtlsample()`, to simplify rolling window estimation of SV models. In this estimation method, either a fixed width time window is *moving* through the time series or a sequence of *expanding* time windows with the same starting time point covers larger and larger chunks of the observations, and the same model is estimated in all time windows independently. Next, each estimated model is employed for out-of-sample prediction, typically one day to one week ahead of the time window. Lastly, the set of predicted values might be used to evaluate the model fit.

In Bayesian statistics, a natural approach for assessing the predictive power of a model is through its *posterior predictive distribution*. Its density, also called the *predictive density*, is defined as

$$p(y_{t+1} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}) = \int_{\boldsymbol{\mathcal{K}}} p(y_{t+1} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}, \boldsymbol{\kappa}) p(\boldsymbol{\kappa} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}) \mathrm{d}\boldsymbol{\kappa}, \tag{12}$$

where $\boldsymbol{\kappa}$ collects all unobserved variables, i.e., $\boldsymbol{\kappa} = (\mu, \varphi, \sigma, \rho, \nu, \boldsymbol{h}, \boldsymbol{\beta})^\top$ in the most general case of SVtl, and the domain of integration $\boldsymbol{\mathcal{K}}$ is the set of all possible values for $\boldsymbol{\kappa}$. We follow Geweke and Amisano (2010) in our notation by using a superscript $\boldsymbol{o}$ for the vector of observed values $\boldsymbol{y}^{\boldsymbol{o}}_{[1:t]} = (y_1, y_2, \ldots, y_t)^\top$. Equation 12 can be seen as the integration of the predictive likelihood over the posterior distribution of all parameters and therefore it accounts for posterior parameter uncertainty for the predicted values.

The integral in Equation 12 has no closed form and its dimensionality increases with t; it is intractable. Hence, we rely on Monte Carlo integration and we simulate from the posterior

predictive distribution. For the evaluation of the predictive density at an observation $x = y^o_{t+1}$, called the *predictive likelihood*, we apply the computation

$$p(x \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}) \approx \frac{1}{M} \sum_{m=1}^{M} p(x \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}, \boldsymbol{\kappa}^{(\boldsymbol{m})}), \tag{13}$$

where $\boldsymbol{\kappa}^{(\boldsymbol{m})}$ denotes the mth posterior sample from the estimation procedure of the SV model. For other applications, the quantiles of the posterior predictive distribution, henceforth the *predictive quantiles*, might be of interest. We estimate the q% quantile through random variates simulated from $y_{t+1} \sim p(y_{t+1} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]})$, which we acquire by repeating two steps for $m = 1, \ldots, M$:

Step 1. Simulate $\boldsymbol{\kappa}^{(\boldsymbol{m})}$ from the SV posterior $p(\boldsymbol{\kappa} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]})$, and

Step 2. Simulate $y^{(m)}_{t+1}$ from $p(y_{t+1} \mid \boldsymbol{y}^{\boldsymbol{o}}_{[1:t]}, \boldsymbol{\kappa}^{(m)})$.

Lastly, we take the q% quantile of the sample vector $(y^{(1)}_{t+1}, y^{(2)}_{t+1}, \ldots, y^{(M)}_{t+1})^\top$ as the approximate q% quantile of the predictive density. We implement the estimation of both the predictive likelihood and predictive quantiles in **stochvol**.

All four rolling window routines `svsample_roll()`, `svtsample_roll()`, `svlsample_roll()`, and `svtlsample_roll()` bear the same programming interface. They expect as their first argument the input data $\boldsymbol{y}_{[1:L]}$, which is of length L. For estimating the SV model in each time window $j = 1, \ldots, J$ in the moving or expanding window scheme, the sub-vector $\boldsymbol{y}_{[j:(t+j-1)]}$, or, respectively, $\boldsymbol{y}_{[1:(t+j-1)]}$, is taken as data and is used to predict $n_{\text{ahead}} \geq 1$ time steps ahead. The width t of the first time window can be determined from L, J, and n_{ahead}. The following example demonstrates how the rolling window sampling routines can be called in **stochvol**.

```
R> set.seed(5)
R> res <- svsample_roll(CHF_logret, n_ahead = 1, forecast_length = 30,
+    refit_window = "moving", calculate_quantile = c(0.01, 0.05),
+    calculate_predictive_likelihood = TRUE)
```

Argument `n_ahead` is used to set n_{ahead}, `forecast_length` is used to set J, and `refit_window` expects either `"moving"` or `"expanding"` to set the rolling window scheme to moving or expanding, respectively. Argument `calculate_quantile` expects a vector of numbers between 0 and 1; the numbers are interpreted as the quantiles to be predicted. Furthermore, if `calculate_predictive_likelihood` is set to `TRUE`, the function estimates the predictive likelihood. Lastly, the output `res` is a list of length J, i.e., one element for each time window. It contains the respective posterior quantile and predictive likelihood results together with all posterior parameter draws for $\boldsymbol{\kappa}$.

4.5. Specifying the prior hyperparameters

As discussed in Section 2.2, the prior distributions need to be specified before the estimation process can start. Concerning the common model parameters μ, φ, and σ, all of

`svsample()`, `svtsample()`, `svlsample()`, and `svtlsample()` expect through their input arguments `priormu`, `priorphi`, and `priorsigma` values for $(b_\mu, \sqrt{B_\mu})$, (a_φ, b_φ), and B_σ, respectively. Furthermore, all sampling functions accept the argument `priorbeta` to set an independent prior for the regression coefficients by providing (b_β, s_β), where b_β and s_β are the common mean and, respectively, the common standard deviation. For a general multivariate normal distribution, the `specify_priors()` interface exists, which we detail later in this Section. The prior for ν can be influenced in `svtsample()` and `svtlsample()` by passing λ_ν as the argument `priornu`. Finally, `svlsample()` and `svtlsample()` take the `numeric` sequence (a_ρ, b_ρ) through the input argument `priorrho`.

The code snippet below shows all the default values of the prior hyperparameters.

```
R> svsample(CHF_logret, priormu = c(0, 100), priorphi = c(5, 1.5),
+    priorsigma = 1, priorbeta = c(0, 10000))
R> svtsample(CHF_logret, priormu = c(0, 100), priorphi = c(5, 1.5),
+    priorsigma = 1, priorbeta = c(0, 10000), priornu = 0.1)
R> svlsample(CHF_logret, priormu = c(0, 100), priorphi = c(5, 1.5),
+    priorsigma = 1, priorbeta = c(0, 10000), priorrho = c(4, 4))
R> svtlsample(CHF_logret, priormu = c(0, 100), priorphi = c(5, 1.5),
+    priorsigma = 1, priorbeta = c(0, 10000), priornu = 0.1,
+    priorrho = c(4, 4))
```

As an alternative to the concise interface above, a broader set of prior distributions can be specified via an object created by the `specify_priors()` function. The function has an input argument for each model parameter: `mu`, `phi`, `sigma2`, `nu`, `rho`, `beta`, and additionally one for the variance of h_0 called `latent0_variance`. There is a list of accompanying functions that create distributions in **stochvol**: `sv_beta()` has arguments `shape1` and `shape2` and it is accepted for `phi` and `rho`; `sv_constant()` has argument `value` and it is accepted for `mu`, `phi`, `sigma2`, `nu`, `rho`, and `latent0_variance`; `sv_normal()` has arguments `mean` and `sd` and it is accepted for `mu` and `phi`; `sv_multinormal()` has arguments `mean` and either `sd` and `dim` or `precision`, and it is accepted for `beta`; `sv_exponential()` has argument `rate` and it accepted for `nu`; `sv_gamma()` has arguments `shape` and `rate` and it is accepted for `sigma2`; `sv_inverse_gamma()` has arguments `shape` and `scale` and it is accepted for `sigma2`; and `sv_infinity()` has no arguments and it is accepted for `nu` hence turning the Student's t distribution into a normal distribution. Additionally, `latent0_variance` accepts the character value `"stationary"`. All four sampling methods accept the prior specification object through the input argument `priorspec`.

All input arguments for `specify_priors` are optional, their default values and how they are used is seen below.

```
R> ps <- specify_priors(mu = sv_normal(mean = 0, sd = 100),
+    phi = sv_beta(shape1 = 5, shape2 = 1.5), rho = sv_constant(0),
+    sigma2 = sv_gamma(shape = 0.5, rate = 0.5), nu = sv_infinity(),
+    beta = sv_multinormal(mean = 0, sd = 10000, dim = 1),
+    latent0_variance = "stationary")
R> svsample(CHF_logret, priorspec = ps)
```

4.6. Setting up the Markov chain

When conducting Bayesian inference using an MCMC sampling scheme, the number of draws from the posterior distribution, the length of the so-called burn-in phase, the initial values of the Markov chain, and the various strategies of storing the results are all of general interest. The input arguments `draws` and `burnin` settle the first two points. A sample size of `burnin + draws` is acquired from the posterior distribution out of which the first `burnin` number of draws are thrown away. The default is to draw 10000 elements after a burnin of 1000 for SV models without leverage, and draw 20000 elements after a burnin of 2000 for SV models with leverage, which in our experience is enough for most applications.

As for the initial values, `startpara` and `startlatent` provide a way to set them. The argument `startpara` is expected to be a named `list` mapping parameter names to starting values, and `startlatent` must be a sequence of length m that contains starting values for $\boldsymbol{h}$. Default values are set to be the prior mean for φ, σ, ν, and ρ, these have only minor influence on the Markov chain. The default value for $\boldsymbol{\beta}$ is the ordinary least squares estimator $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$, where $\boldsymbol{X}$ denotes the regression design matrix and $\boldsymbol{y}$ denotes the vector of observations. After setting $\boldsymbol{\beta}$, the level of log-variance μ is initialized according to the Bayesian linear regression

$$\begin{aligned} \log(y_t^2) &= \mu + \xi_t, \\ \mu &\sim \mathcal{N}(b_\mu, B_\mu), \end{aligned} \tag{14}$$

where $\xi_t \sim \mathcal{N}(-1.27, 4.934)$. Equation 14 results from the first line of Equation 1 by fixing h_t at its stationary expected value μ and then taking $x \mapsto \log(x^2)$ of both sides. The homoskedastic error term ξ_t is acquired as the Laplace approximation to $\log(\varepsilon_t^2)$ (Harvey and Shephard 1996). At the end, by default all values of the vector `startlatent` are set to the initial value of μ.

It is customary to start independent Markov chains in parallel and **stochvol** provides facilities for that in all of its sampling procedures. The argument `n_chains` is expected to be a positive integer, it sets the number of independent chains. Additionally, arguments `parallel`, `n_cpus`, and `cl` can be used to control parallelism used by **stochvol**. To overwrite the default sequential execution strategy, `parallel` is to be set either to `"snow"`, to employ the so-called "SNOW" clusters, or to `"multicore"` to use the "multicore" type computation (R Core Team 2021). Next, argument `n_cpus` should be set to the physical number of parallel processing units to be used. Finally, in case "SNOW" is applied, the sampling routines optionally accept an already running "SNOW" cluster through argument `cl`.

As mentioned earlier, the sampling algorithms for the latent states $\boldsymbol{h}$ in **stochvol** rely on a Gaussian mixture approximation as in Omori *et al.* (2007) and Kastner and Frühwirth-Schnatter (2014). The approximation tends to be very good, therefore the default setting is not to correct for model misspecification. However, this correction can be enabled in all of the sampling routines through the `expert` argument as shown for `svsample()` in the following.

```
R> set.seed(3)
R> y <- svsim(50)$y
R> svsample(y, expert = list(correct_model_misspecification = TRUE))
```

Lastly, **stochvol** provides three ways to economize storage during and after the execution of the sampler. Setting the `integer` argument `thinpara` to ι tells the sampler to store only

every ιth draw of the vector of parameters, and supplying a value for `thinlatent` does the same for $\boldsymbol{h}$. Finally, one has the opportunity not to store the full vector $\boldsymbol{h}$ but only its last value by setting `keeptime = "last"`. The default behavior is to store every draw after the burn-in phase.

5. The factorstochvol package

The most common workflow of using **factorstochvol** for fitting multivariate factor SV models consists of the following steps: (1) Prepare the data, (2) decide on an identification structure, (3) specify the prior hyperparameters, (4) run the sampler, (5) investigate the output and visualize the results, and (6) predict (if required). These steps are described in detail in the following sections.

5.1. Preparing the data

The workhorse in **factorstochvol** is the sampling function `fsvsample()`. It expects the data to come in form of a matrix $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)^\top$ with n rows and m columns. For illustration, we use the exchange rate data set in **stochvol** which contains 3140 daily observations of exchanges rates for 23 currency pairs against EUR, ranging from March 3, 2000 to April 4, 2012. To keep the analysis simple and computation times moderate, we however only model the last 201 days of the first six series in alphabetical order (Australian dollar, Canadian dollar, Swiss franc, Czech koruna, Danish krone, Great British pound) for further analysis. Instead of using the nominal exchange rates we compute log returns. This leaves us with a data set of size $n = 200$ and $m = 6$.[7] The data is prepared using the code snippet below and visualized in Figure 4 using the **zoo** package (Zeileis and Grothendieck 2005).

```
R> library("factorstochvol")
R> library("zoo")
R> data("exrates", package = "stochvol")
R> m <- 6
R> n <- 200
R> y <- 100 * logret(tail(exrates[, seq_len(m)], n + 1))
R> y <- zoo(y, order.by = tail(exrates$date, n))
R> plot(y, main = "", xlab = "Time")
```

5.2. Deciding on an identification structure

The likelihood in factor models is invariant to certain factor transformations such as reordering of factors and their loadings or sign switches thereof. In addition to this, it is often multimodal. Consequently, identifying the factor loadings is far from trivial. The most common way to address this issue in factor SV models is to impose a lower-diagonal factor loadings matrix where all elements above the diagonal are set to zero (e.g., Aguilar and West 2000; Chib *et al.* 2006; Han 2006; Zhou, Nakajima, and West 2014). To use this constraint in **factorstochvol**, the argument `restrict = "upper"` can be passed to the main sampling function `fsvsample()`.

[7] In the original paper, $n = 1000$ is used.

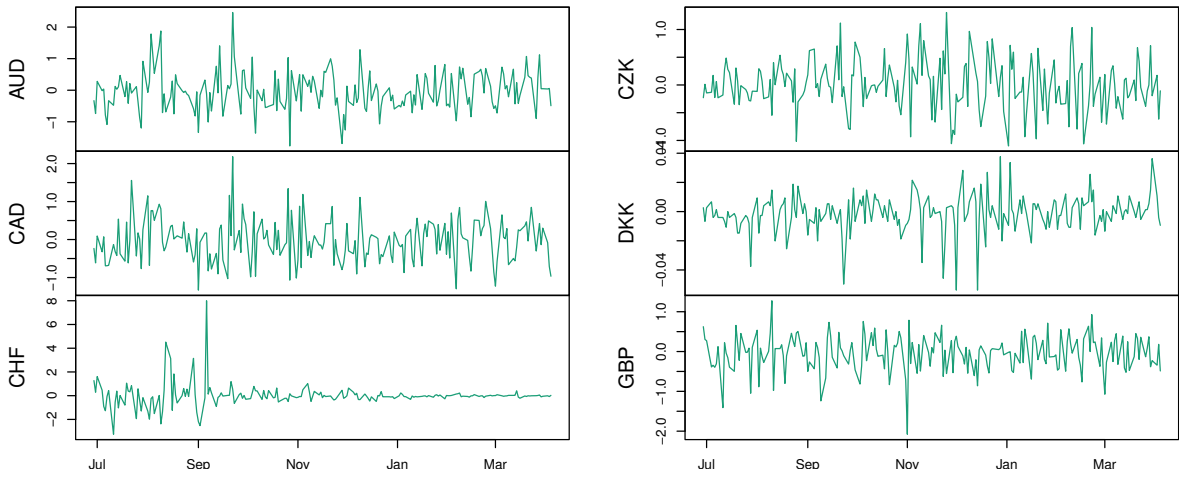

Figure 4: Percentage log returns of six EUR exchange rates.

Evidently, this practice imposes an order dependence, as, e.g., the first variable is not allowed to load on anything else but the first factor.

A rather ad hoc method for automatically ordering the data is implemented in the helper function `preorder()`. After a maximum likelihood factor model fit to the data (using `factanal()` from the **stats** package with the default `varimax` rotation), the series are ordered as follows: The variable with the highest loading on factor 1 is placed first, the variable with the highest loading on factor 2 second (unless this variable is already placed first, in which case the variable with the second highest loading is taken), et cetera. For the data set at hand, this would imply the following ordering for a two-factor model.

```
R> preorder(y, factors = 2)
```

According to this algorithm, the second series should be placed first and the third series should be placed second. Thereafter, the alphabetical ordering remains.

To achieve this effect without reordering the data, a `logical` matrix of size $m \times r$ can be passed to `fsvsample()` via `restrict`, where the entry `TRUE` means that this element is restricted to zero; `FALSE` means that it is to be estimated from the data. Similarly to `preorder()`, the function `findrestrict()` tries to automate this procedure. Again, the maximum likelihood estimates from a static factor analysis are used; however, `findrestrict()` uses a slightly different algorithm than the one above: The variable with the lowest absolute loadings on factors $2, 3, \ldots, r$ (relative to factor 1) is determined to lead the first factor, the variable with the lowest absolute loadings on factors $3, 4, \ldots, r$ (relative to factors 1 and 2) is placed second, et cetera. Below is the result for the data set at hand.

```
R> findrestrict(y, factors = 2)
```

If `fsvsample()` is called with the argument `restrict = "auto"`, it automatically invokes `findrestrict()` with the appropriate number of factors. Using `restrict = "none"` (the default) causes the sampler not to place any constraints on the loadings matrix; thus, the resulting posterior draws may be unstable or suffer from multiple local modes. If, however, inference on the factor loadings themselves is not the primary concern of the analysis, leaving

the factor loadings unidentified may be the preferred option. This is in particular the case when inference for the covariance matrix is sought, as this only depends on $\mathbf{\Lambda}$ through the rotation-invariant transformation of Equation 9. For a more elaborate discussion of these issues, we refer the reader to Sentana and Fiorentini (2001) who discuss automatic identification through heteroskedasticity. A comparison of log predictive scores under different identification schemes for factor SV models is given in Kastner *et al.* (2017); see also Frühwirth-Schnatter and Lopes (2018) for related issues in static factor models. To continue with the current example, we chose not to place any a priori restrictions on the factor loadings matrix while using a row-wise normal-gamma shrinkage prior on the factor loadings matrix (cf. Kastner 2019).

5.3. Specifying prior hyperparameters

Apart from the obvious prior choice about the number of factors and the identification scheme discussed above, a number of hyperparameter choices are available in **factorstochvol**. Regarding the log-variance processes, the interface is analogous to that of `svsample()`. In the following, $i = 1, \ldots, m$ and $j = 1, \ldots, r$ index the idiosyncratic and the factor log-variance processes, respectively. The pair of common prior hyperparameters (b_β, B_β) can be passed as a sequence of length two to `priorbeta`. The common prior of $\bar{\mu}_i$ can be set by passing a sequence of length two – the mean and the standard deviation of the normal distribution – to `priormu`; the common priors of $\bar{\varphi}_i$ and $\tilde{\varphi}_j$ can also be set by passing sequences of length two – the parameters of the corresponding beta distribution – to `priorphiidi` and to `priorphifac`, respectively; similarly, the common priors of $\bar{\sigma}_i$ and $\tilde{\sigma}_j$ can be specified via the arguments `priorsigmaidi` and `priorsigmafac`, respectively, that accept as positive numbers the scale B_σ of the corresponding gamma distribution.

As discussed in Section 3.2, **factorstochvol** offers three specifications as priors for $\mathbf{\Lambda}$, controlled through the argument `priorfacloadtype`. To use the first option (`priorfacloadtype = "normal"`), one needs to fix the values of τ_{ij} a priori. The user can pass these fixed values to `fsvsample()` via the argument `priorfacload`, either as an $m \times r$ matrix with positive entries or as a single positive number which will be recycled accordingly. For the second option, the normal gamma prior with row-wise or column-wise shrinkage (`priorfacloadtype = "rowwiseng"` and `priorfacloadtype = "colwiseng"`, respectively), the value of argument `priorfacload` is then interpreted as the shrinkage parameter a. Both specifications of the normal gamma prior need the values c and d. They can be set as a two-element vector passed to the argument `priorng`.

5.4. Running the MCMC sampler

Running the sampler corresponds to invoking `fsvsample()`. Apart from the prior settings discussed above, its most important arguments are listed below with the default value in brackets. For a complete list of all arguments and more details, see `?fsvsample`.

- `y`: the data;
- `factors` [1]: the number of factors;
- `draws` [1000]: the number of MCMC samples to be drawn after burnin;
- `thin` [1]: the amount of thinning (every `thin`th draw is kept);

- `burnin` [1000]: the length of the burn-in period, i.e., the number of MCMC draws to be discarded before the samples are considered to emerge from the stationary distribution,
- `zeromean` [`TRUE`]: a logical value indicating whether β is to be estimated from the data or whether β is set to zero (the default);
- `keeptime` [`"last"`]: either `"all"`, meaning that all latent log volatilities are being monitored at all points in time, or `"last"`, meaning that the latent log volatility draws are only stored at $t = n$, the last point in time; the latter setting is the default to avoid excessive memory usage in higher dimensions;
- `heteroskedastic` [`TRUE`]: indicator(s) to turn off stochastic volatility for the idiosyncratic variances, the factor variances, or both;
- `samplefac` [`TRUE`]: indicator to turn off sampling of the factors; useful to work with observed instead of latent factors (see Kastner 2019, for a use case of this);
- `runningstore` [6]: to avoid having to store all MCMC draws, `fsvsample`'s default is to compute and store the first two ergodic moments of some interesting quantities (namely log variances, factors, volatilities, covariance matrices, correlation matrices, communalities) only; the default (`runningstore = 6`) is to compute and store everything; however, one can set `runningstore` to a lower number to save computation time; the argument `runningstoremoments` [2] can further be used to modify the number of moments to be stored;
- `runningstorethin` [10]: indicates how often ergodic moments should be calculated, where 1 means that this should be done at every iteration and higher numbers lessen both runtime as well as accuracy;
- `quiet` [`FALSE`]: a logical indicator determining the verbosity of `fsvsample`.

For our illustrative example, most settings are left at their default values. The number of factors is increased from one to two, instead of 1000 we sample 3000 draws[8], we estimate a constant mean, a thinning of 10 is used, and `quiet` is set to `TRUE`.

```
R> set.seed(1)
R> res <- fsvsample(y, factors = 2, draws = 3000, zeromean = FALSE,
+    thin = 10, quiet = TRUE)
```

5.5. Investigating the output and visualizing the results

The resulting object

```
R> res
```

holds a rich amount of information. In particular, it contains

[8] In the original paper, 10000 draws are sampled.

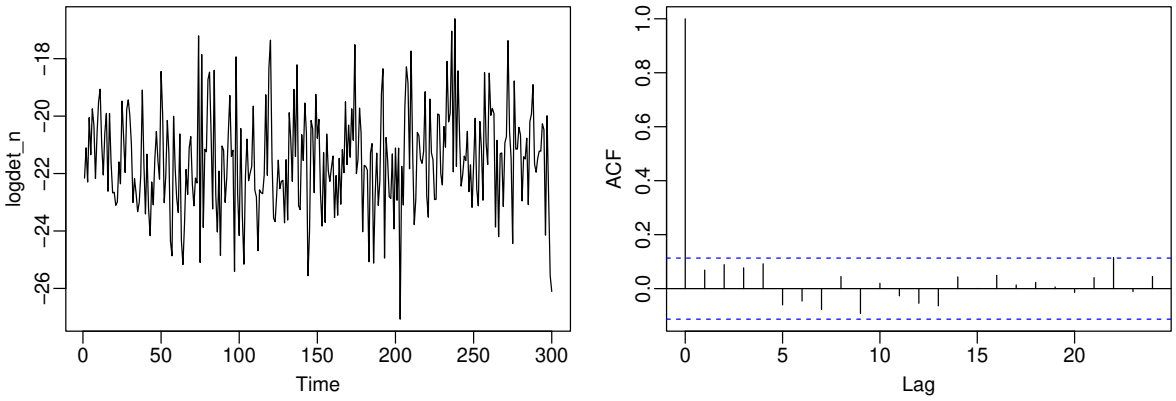

Figure 5: Trace plot and empirical autocorrelation function of the log determinant of the model-implied covariance matrix at $t = n$.

- draws of certain posterior quantities such as the factors $\boldsymbol{f}$, the factor loadings $\boldsymbol{\Lambda}$, the various factor and idiosyncratic SV parameters, the latent factor and idiosyncratic log variances $\tilde{\boldsymbol{h}}$ and $\bar{\boldsymbol{h}}$, and the intercept $\boldsymbol{\beta}$,
- configuration settings such as the number of draws, potential restrictions on the loadings matrix, prior hyperparameters, etc.,
- running moments (such as means and standard deviations) of quantities of interest, depending on the values of `runningstore` and `runningstoremoments` specified when calling `fsvsample()`,
- the data input $\boldsymbol{y}$.

For more details, please investigate `str(res)` and/or `?fsvsample`.

Using `covmat()`, one can extract the MCMC draws of the implied covariance matrices for all points in time which have been stored during sampling. By default, this is the last point in time (`keeptime = "last"`), and thus

```
R> dim(cov_n <- covmat(res))
```

shows that we have stored 300 posterior draws of a 6×6 covariance matrix at one point in time, $t = n = 200$. To check convergence, one can take a look at the trace plot and the autocorrelation function of the log determinant, i.e.,

```
R> logdet <- function (x) log(det(x))
R> logdet_n <- apply(cov_n[,,,1], 3, logdet)
R> ts.plot(logdet_n)
R> acf(logdet_n, main = "")
```

The results are visualized in Figure 5; decent mixing for this quantity is apparent.

To assess the mixing speed of each individual covariance matrix element, one can check, e.g., the estimated effective sample size (out of 300 draws kept) which is implemented in **coda**. Again, no major convergence problems are apparent.

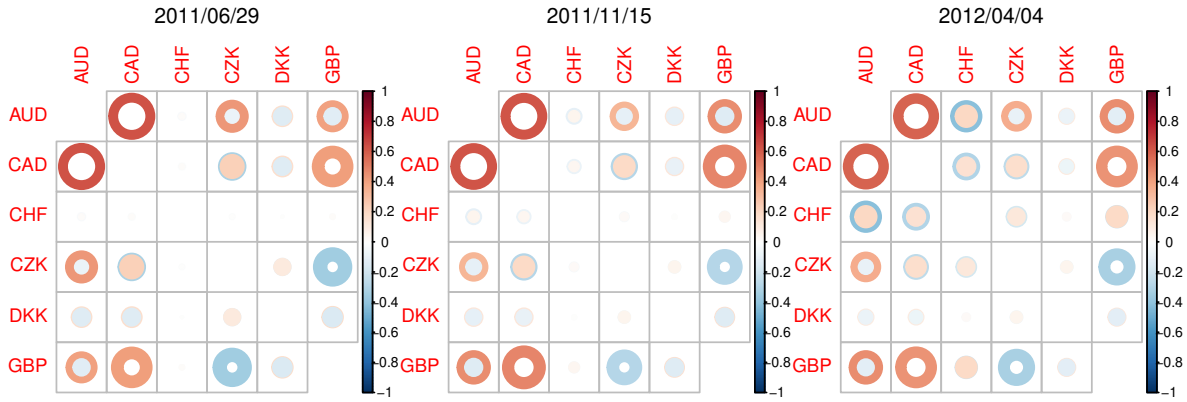

Figure 6: Three estimated correlation matrices and their posterior uncertainty depicted using circles. Inner (outer) radii of the circles illustrate to the posterior mean minus (plus) two standard deviations. Colors blue and red represent negative and positive values, respectively. Furthermore, the transparency of the circles represents the posterior mean. Note that the diagonal is left white as it trivially contains ones (without uncertainty).

```
R> round(apply(cov_n, 1:2, coda::effectiveSize))
```

Assuming that `runningstore` was set sufficiently high when sampling, several convenience functions can be used for quick visualizations without having to post-process the MCMC draws. For example, to visualize the time-varying correlation matrices, consider

```
R> corimageplot(res, these = seq(1, n, length.out = 3), plotCI = "circle",
+    plotdatedist = 2, date.cex = 1.1)
```

which produces the three estimated posterior correlation matrices depicted in Figure 6. Setting `plotCI = "circle"` visualizes posterior uncertainty – inner and outer radii correspond to the posterior mean plus/minus two standard deviations, respectively.

To get an idea about how the marginal volatilities evolve over time, `voltimeplot()` can be used. To exemplify,

```
R> palette(RColorBrewer::brewer.pal(7, "Dark2")[-5])
R> voltimeplot(res, legend = "top")
```

yields the estimated volatilities in Figure 7. The financial crisis of 2008 and the capping of CHF's appreciation in September 2011 are clearly visible, while DKK's volatility (relative to EUR) is practically zero. Note that `voltimeplot()` respects palette changes. In the above example, **RcolorBrewer** (Neuwirth 2014) is used. Moreover,

```
R> palette(RColorBrewer::brewer.pal(6, "Dark2"))
R> cortimeplot(res, 1)
R> cortimeplot(res, 2)
```

yields the estimated pairwise correlations in Figure 8. While, relative to EUR, the estimated correlation between AUD and CAD appears to be relatively stable over time, correlations with

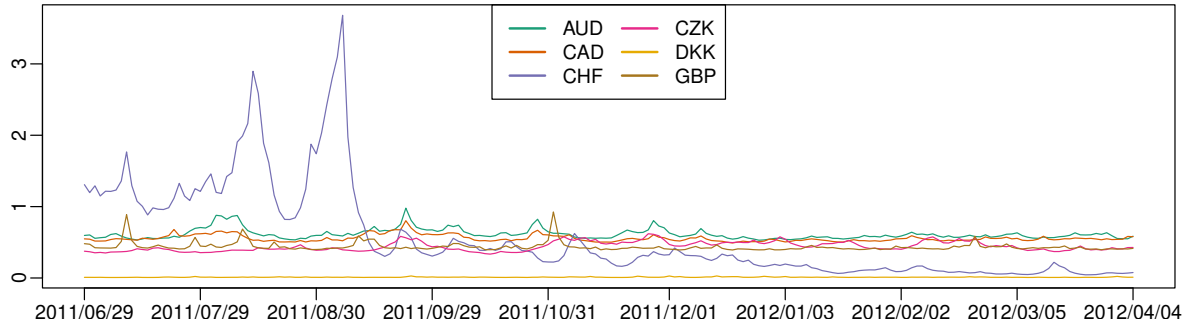

Figure 7: Posterior means of daily marginal volatilities in percent.

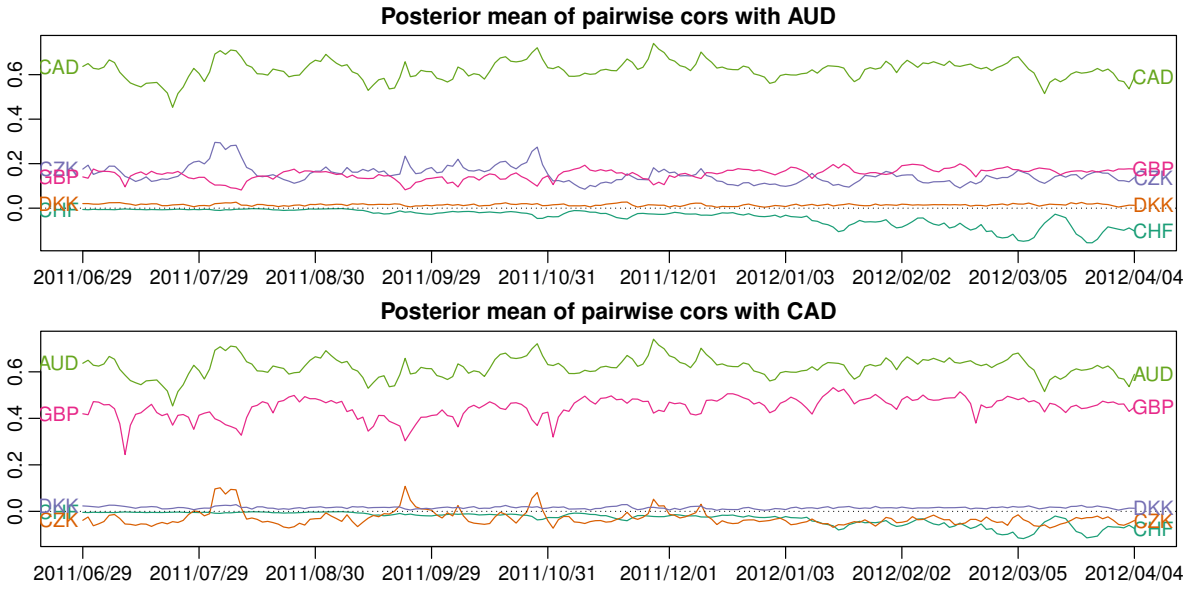

Figure 8: Posterior means of correlations with AUD (top panel) and CAD (bottom panel).

CHF can become negative at times. To visualize the *communalities*, i.e., the proportions of variances explained through the latent factors, invoke

```
R> comtimeplot(res, maxrows = 6)
```

which yields the estimated communalities in Figure 9.

To gain an even deeper understanding of the estimated model, we now turn towards examining the latent factors and their variances themselves. To visualize the loadings, the functions `facloadpairplot()`, `facloadcredplot()`, `facloadpointplot()`, `facloadtraceplot()`, and `facloaddensplot()` are available; the former two are exemplified in Figure 10. Moreover, we can see the factor log variances produced through `logvartimeplot(res, show = "fac")`. Similarly, `logvartimeplot(res, show = "idi")` produces plots of the idiosyncratic log variances which are displayed in Figure 11.

```
R> facloadpairplot(res)
R> facloadcredplot(res)

R> logvartimeplot(res, show = "fac")

R> logvartimeplot(res, show = "idi", maxrows = 6)
```

In addition to the above, there is the plotting function `paratraceplot()` which produces trace plots of all parameters associated with the log variances processes: mean, persistence, and volatility of log variances.

In order to provide some guidance when it comes to selecting the number of factors, **factorstochvol** ships with the function `evdiag()`. It computes and visualizes the eigenvalues of $\mathbf{\Lambda}^\top \mathbf{\Lambda}$ which can be used as a rough guide in analogy to a scree plot for static factor models. To use it, one can fit a model with a relatively large number of factors and assess the importance of each of these (in descending order). For example, the code below produces Figure 12, hinting at a model with no more than four factors.

```
R> set.seed(6)
R> largemodel <- fsvsample(y, factors = 6)
R> evdiag(largemodel)
```

5.6. Predicting covariances, correlations, and future observations

One of the main use cases of **factorstochvol** might be to predict covariance and correlation matrices of time series. To this end, `predcov()` and `predcor()` yield draws from the posterior predictive distribution of these, defined in analogy to Equation 12. For instance, the code below can be used to obtain one-step-ahead posterior predictive means and standard deviations for the correlation matrix on April 5, 2012 (using data up to April 4 only).

```
R> set.seed(4)
R> predcor1 <- predcor(res)
R> round(apply(predcor1[,,,1], 1:2, mean), 2)
R> round(apply(predcor1[,,,1], 1:2, sd), 2)
```

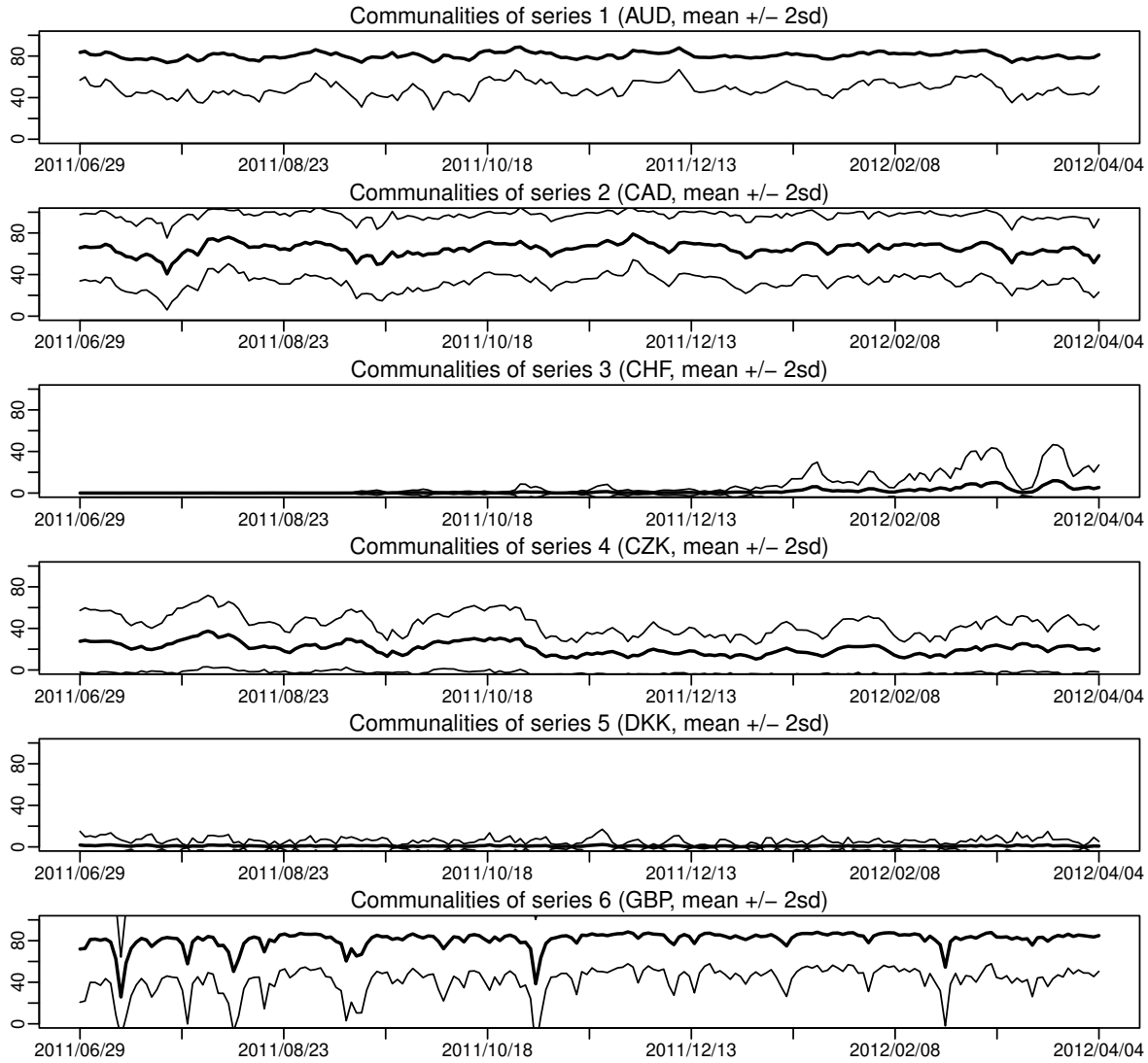

Figure 9: Communalities: Posterior means plus/minus two posterior standard deviations. The six panels correspond to the six observation series and they depict the percentage of volatility explained by the latent factors. The idiosyncratic volatility attributes for the unexplained part.

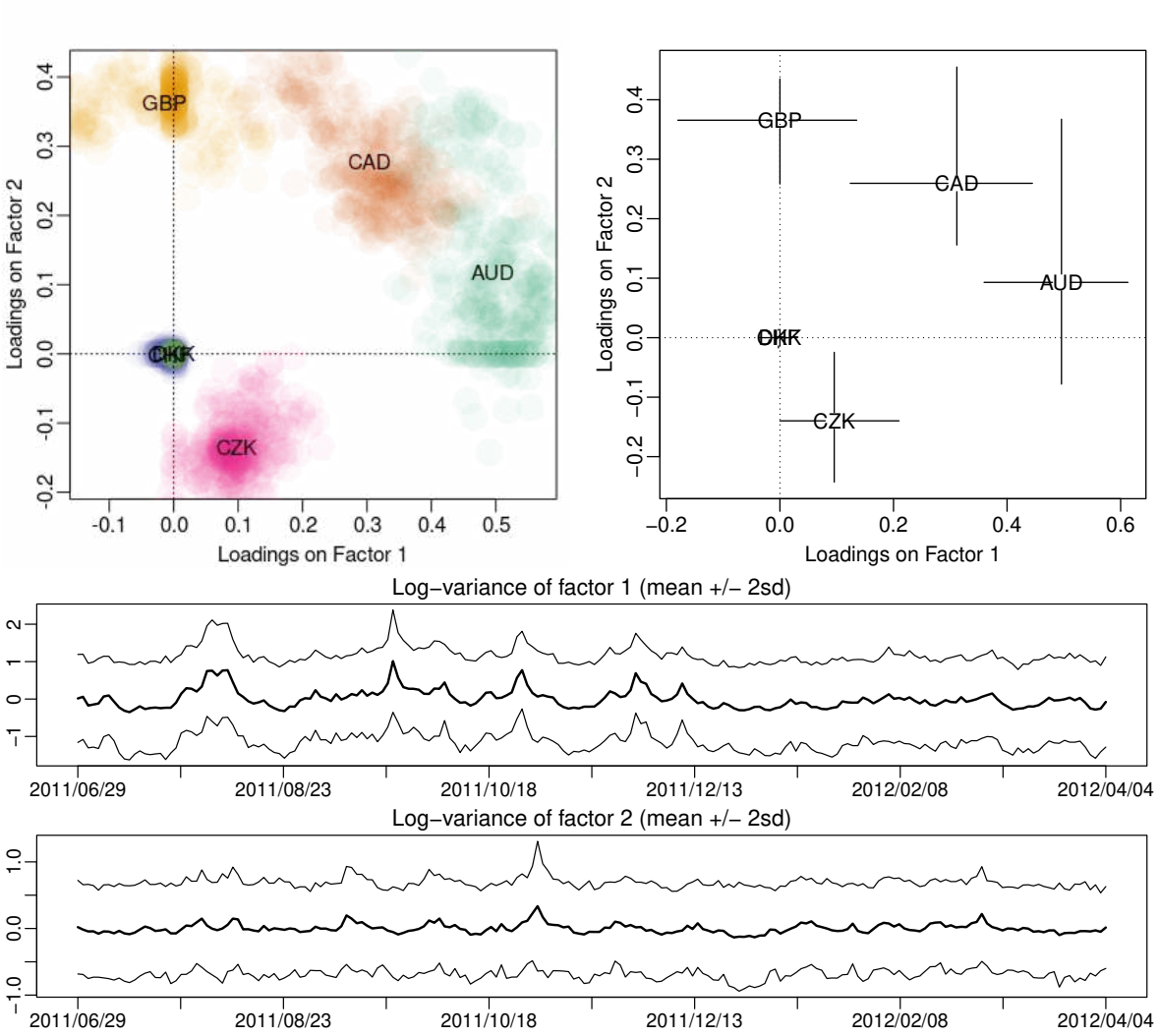

Figure 10: Factor loadings and factor variances. The top panels visualize the joint posterior distribution of the two factor loadings. While the colored clouds on the left consist of posterior draws and provide details, the posterior 0.01/0.99 credible intervals on the right summarize the marginal distributions. The bottom panels depict posterior means plus/minus two standard deviations of factor log variances.

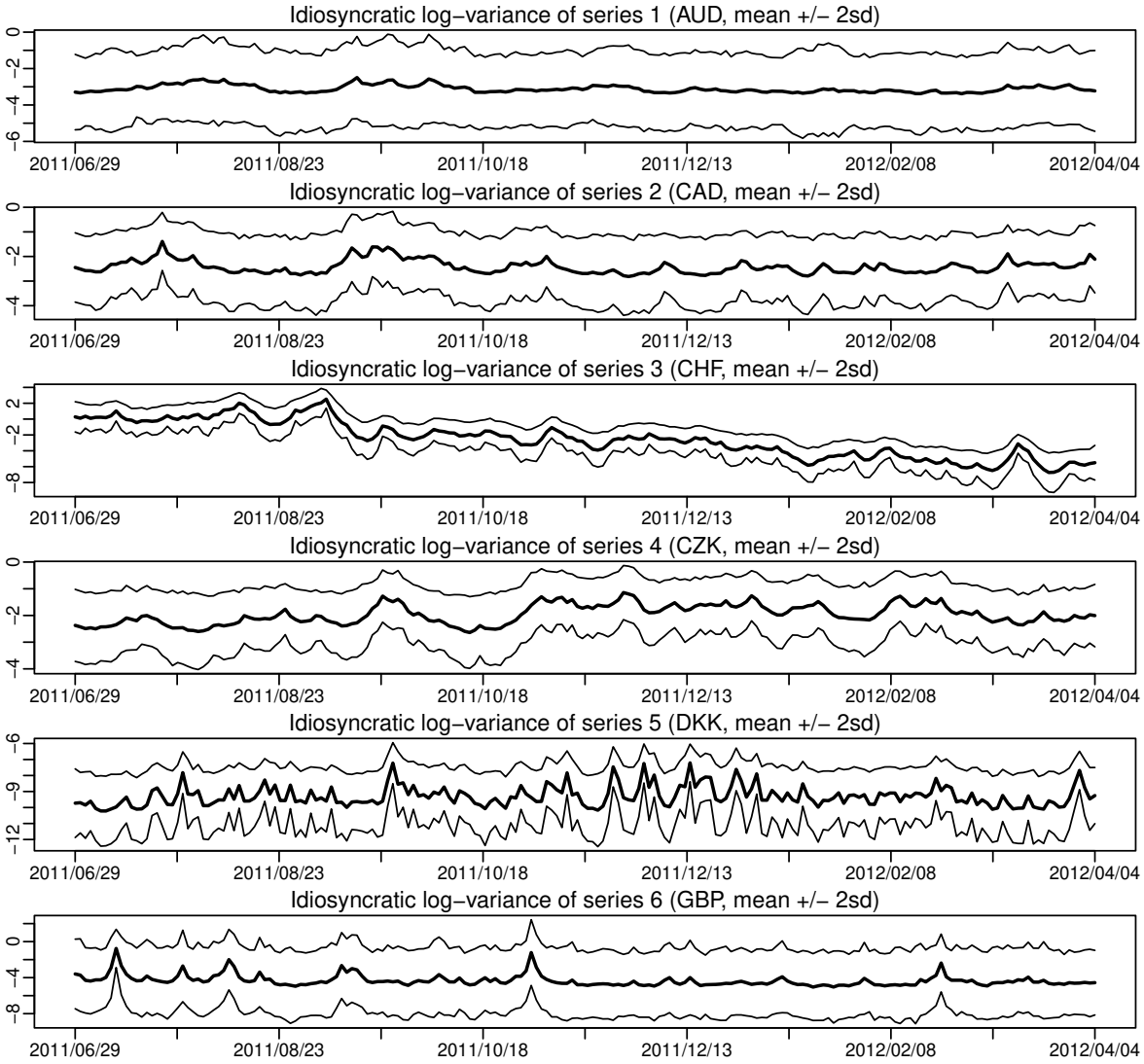

Figure 11: Idiosyncratic log variances: Posterior means plus/minus two standard deviations. Panels illustrate the marginal posterior distributions of the diagonal elements of $\bar{\mathbf{\Sigma}}_t$ in Equation 8.

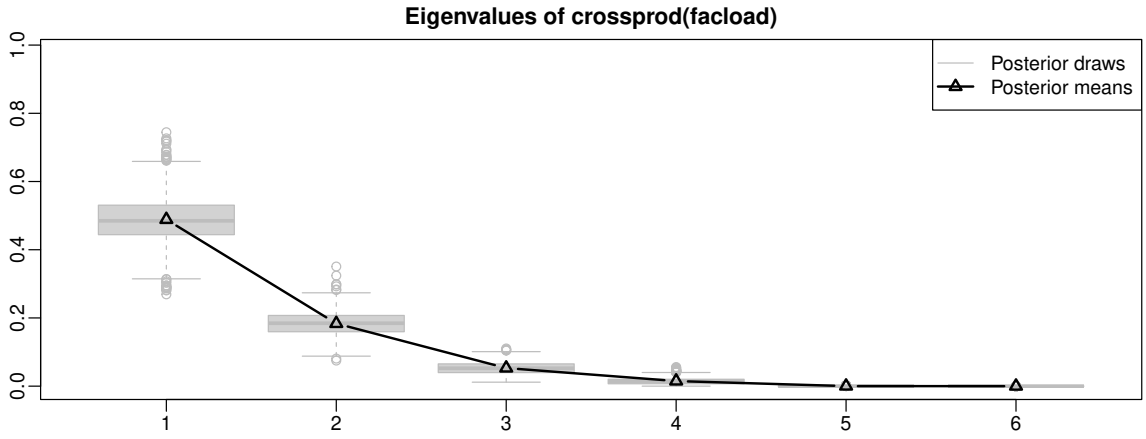

Figure 12: Eigenvalues of $\mathbf{\Lambda}^\top \mathbf{\Lambda}$ which can be used as a rough guide to selecting the number of factors.

To obtain draws from the posterior predictive distribution of new data points, one can simply draw from the corresponding mixture of multivariate normals. In Figure 13, these draws are visualized via `heatpairs()` from **LSD** (Schwalb, Tresch, Torkler, Duemcke, Demel, Ripley, and Venables 2018).

```
R> set.seed(5)
R> predcov_1 <- predcov(res)
R> preddraws <- t(res$beta)
R> for (i in seq_len(NROW(preddraws)))
+    preddraws[i,] <- preddraws[i,] + chol(predcov_1[,,i,1]) %*% rnorm(m)
R> plotlims <- quantile(preddraws, c(0.01, 0.99))
R> LSD::heatpairs(preddraws, labels = colnames(y), cor.cex = 1.5, gap = 0.3,
+    xlim = plotlims, ylim = plotlims)
```

To conclude, we note that convenience functions such as `predloglik()` or `predloglikWB()` may aid in approximating predictive likelihoods (cf., e.g., Geweke and Amisano 2010). For instance, assuming that the actually observed value of $\boldsymbol{y}_{n+1} = \boldsymbol{y}_{n+2} = (0, 0, 0, 0, 0, 0)^\top$, we can approximate the one and two step ahead log predictive scores through code along the following lines.

```
R> set.seed(6)
R> predloglik(res, matrix(0, nrow = 2, ncol = m), ahead = 1:2, each = 10)
```

For a more detailed analysis of the out-of-sample performance of **factorstochvol** for exchange rate data, we refer to Kastner *et al.* (2017); for a predictive exercise on stock market data, please see Kastner (2019).

6. Summary and discussion

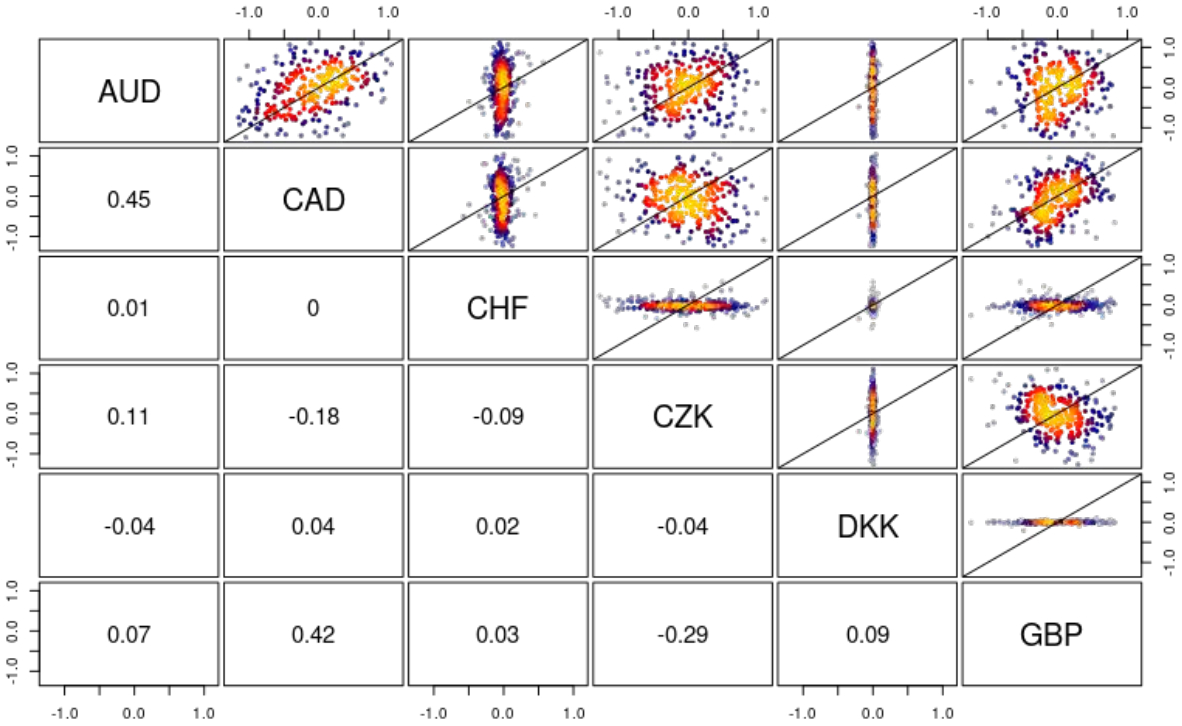

Figure 13: Predictive draws from the one-step-ahead posterior predictive distribution (above the diagonal) and empirical correlation coefficients (below the diagonal).

We extended the work of Kastner (2016) to other SV models, including the univariate heavy-tailed SV, the SV model with leverage, and the multivariate factor SV model. We showcased the features that are the most important to end users in R: estimation through the sampler functions, visualization, summary, and prediction methods. Due to its more involved nature, however, we did not include the description of the C++ interface. Two functions called `update_fast_sv()` and `update_general_sv()` are exported and programmers have the possibility to access the samplers in **stochvol** directly from C++ after linking to the compiled package. For usage examples, see the implementations of **factorstochvol** or **shrinkTVP** (Knaus, Bitto-Nemling, Cadonna, and Frühwirth-Schnatter 2020). Source code is available in the GitHub repositories `https://github.com/gregorkastner/stochvol` and `https://github.com/gregorkastner/factorstochvol`.

Acknowledgments

Gregor Kastner acknowledges funding from the Austrian Science Fund (FWF) for the project "High-dimensional statistical learning: New methods to advance economic and sustainability policy" (ZK 35), jointly carried out by the University of Klagenfurt, WU Vienna University of Economics and Business, Paris Lodron University Salzburg, TU Wien, and the Austrian Institute of Economic Research (WIFO).

References

Abanto-Valle CA, Langrock R, Chen MH, Cardoso MV (2017). "Maximum likelihood estimation for stochastic volatility in mean models with heavy-tailed distributions." *Applied Stochastic Models in Business and Industry*, **33**(4), 394–408. doi:10.1002/asmb.2246.

Aguilar O, West M (2000). "Bayesian Dynamic Factor Models and Portfolio Allocation." *Journal of Business and Economic Statistics*, **18**(3), 338–357. doi:10.1080/07350015.2000.10524875.

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

Bos CS (2012). "Relating Stochastic Volatility Estimation Methods." In L Bauwens, C Hafner, S Laurent (eds.), *Handbook of Volatility Models and Their Applications*, pp. 147–174. John Wiley & Sons. doi:10.1002/9781118272039.ch6.

Brooks S, Gelman A, Jones G, Meng XL (eds.) (2011). *Handbook of Markov Chain Monte Carlo*. Handbooks of Modern Statistical Methods, 1st edition. Chapman and Hall, CRC Press. ISBN 9781420079418.

Carter CK, Kohn R (1994). "On Gibbs Sampling for State Space Models." *Biometrika*, **81**(3), 541–553. doi:10.1093/biomet/81.3.541.

Chib S, Nardari F, Shephard N (2006). "Analysis of High Dimensional Multivariate Stochastic Volatility Models." *Journal of Econometrics*, **134**(2), 341–371. doi:10.1016/j.jeconom.2005.06.026.

Creal DD (2017). "A Class of Non-Gaussian State Space Models With Exact Likelihood Inference." *Journal of Business & Economic Statistics*, **35**(4), 585–597. doi:10.1080/07350015.2015.1092977.

Delatola EI, Griffin JE (2011). "Bayesian Nonparametric Modeling of the Return Distribution with Stochastic Volatility." *Bayesian Analysis*, **6**, 901–926. doi:10.1214/11-BA632.

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 and Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

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

Frühwirth-Schnatter S (1994). "Data Augmentation and Dynamic Linear Models." *Journal of Time Series Analysis*, **15**(2), 183–202. doi:10.1111/j.1467-9892.1994.tb00184.x.

Frühwirth-Schnatter S, Lopes HF (2018). "Sparse Bayesian Factor Analysis when the Number of Factors is Unknown." *arXiv e-prints*, arXiv:1804.04231. URL https://arxiv.org/abs/1804.04231.

Frühwirth-Schnatter S, Wagner H (2010). "Stochastic Model Specification Search for Gaussian and Partial Non-Gaussian State Space Models." *Journal of Econometrics*, **154**(1), 85–100. doi:10.1016/j.jeconom.2009.07.003.

Geweke J (1993). "Bayesian Treatment of the Independent Student-t Linear Model." *Journal of Applied Econometrics*, **8**(S1), S19–S40. doi:10.1002/jae.3950080504.

Geweke J (2004). "Getting It Right: Joint Distribution Tests of Posterior Simulators." *Journal of the American Statistical Association*, **99**, 799–804.

Geweke J, Amisano G (2010). "Comparing and Evaluating Bayesian Predictive Distributions of Asset Returns." *International Journal of Forecasting*, **26**(2), 216–230. doi:10.1016/j.ijforecast.2009.10.007.

Ghalanos A (2020). ***rugarch**: Univariate GARCH Models*. R package version 1.4-4, URL https://CRAN.R-project.org/package=rugarch.

Ghysels E, Harvey AC, Renault E (1996). "Stochastic Volatility." In GS Maddala, CR Rao (eds.), *Handbook of Statistics*, volume 14, chapter 5, pp. 119–191. Elsevier. doi:10.1016/s0169-7161(96)14007-4.

Griffin JE, Brown PJ (2010). "Inference with Normal-Gamma Prior Distributions in Regression Problems." *Bayesian Analysis*, **5**(1), 171–188. doi:10.1214/10-BA507.

Han Y (2006). "Asset Allocation with a High Dimensional Latent Factor Stochastic Volatility Model." *Review of Financial Studies*, **19**(1), 237–271. doi:10.1093/rfs/hhj002.

Harvey AC, Ruiz E, Shephard N (1994). "Multivariate Stochastic Variance Models." *The Review of Economic Studies*, **61**(2), 247–264. doi:10.2307/2297980.

Harvey AC, Shephard N (1996). "Estimation of an Asymmetric Stochastic Volatility Model for Asset Returns." *Journal of Business & Economic Statistics*, **14**(4), 429–434. doi:10.1080/07350015.1996.10524672.

Higham NJ (1990). "Analysis of the Cholesky Decomposition of a Semi-Definite Matrix." *Technical report*, Manchester Institute for Mathematical Sciences. MIMS EPrint 2008.56, URL http://eprints.maths.manchester.ac.uk/1193/.

Hosszejni D, Kastner G (2019). "Approaches Toward the Bayesian Estimation of the Stochastic Volatility Model with Leverage." In R Argiento, D Durante, S Wade (eds.), *Bayesian Statistics: New Challenges and New Generations*, pp. 75–83. Springer-Verlag. doi:10.1007/978-3-030-30611-3_8.

Hosszejni D, Kastner G (2021a). "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.

Hosszejni D, Kastner G (2021b). ***stochvol**: Efficient Bayesian Inference for Stochastic Volatility (SV) Models*. R package version 3.2.0, URL http://CRAN.R-project.org/package=stochvol.

Huber ML (2015). *Perfect Simulation*. Monographs on Statistics and Applied Probability, 1st edition. Chapman and Hall, CRC Press. ISBN 9781482232448.

ISO/IEC (2017). *International Standard ISO/IEC 14882:2017(E) – Programming Language C++*. International Organization for Standardization (ISO). URL https://isocpp.org/std/the-standard.

Jacquier E, Polson NG, Rossi PE (1994). "Bayesian Analysis of Stochastic Volatility Models." *Journal of Business & Economic Statistics*, **20**(1), 69–87. doi:10.1080/07350015.1994.10524553.

Jacquier E, Polson NG, Rossi PE (2004). "Bayesian Analysis of Stochastic Volatility Models with Fat-Tails and Correlated Errors." *Journal of Econometrics*, **122**(1), 185–212. doi:10.1016/j.jeconom.2003.09.001.

Jensen MJ, Maheu JM (2010). "Bayesian Semiparametric Stochastic Volatility Modeling." *Journal of Econometrics*, **157**(2), 306–316. doi:10.1016/j.jeconom.2010.01.014.

Jensen MJ, Maheu JM (2014). "Estimating a Semiparametric Asymmetric Stochastic Volatility Model with a Dirichlet Process Mixture." *Journal of Econometrics*, **178**(3), 523–538. doi:0.1016/j.jeconom.2013.08.018.

Kastner G (2015). "Heavy-Tailed Innovations in the R Package **stochvol**." *Technical report*, WU Vienna University of Economics and Business. URL https://epub.wu.ac.at/4918.

Kastner G (2016). "Dealing with Stochastic Volatility in Time Series Using the R Package **stochvol**." *Journal of Statistical Software*, **69**(5), 1–30. doi:10.18637/jss.v069.i05.

Kastner G (2019). "Sparse Bayesian Time-Varying Covariance Estimation in Many Dimensions." *Journal of Econometrics*, **210**(1), 98–115. doi:10.1016/j.jeconom.2018.11.007.

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. 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**(4), 905–917. doi:10.1080/10618600.2017.1322091.

Kastner G, Hosszejni D (2021). ***factorstochvol**: Bayesian Estimation of (Sparse) Latent Factor Stochastic Volatility Models*. R package version 1.0.0, URL https://cran.r-project.org/package=factorstochvol.

Kim S, Shephard N, Chib S (1998). "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models." *The Review of Economic Studies*, **65**(3), 361–393. doi:10.1111/1467-937x.00050.

Knaus P, Bitto-Nemling A, Cadonna A, Frühwirth-Schnatter S (2020). ***shrinkTVP**: Efficient Bayesian Inference for Time-Varying Parameter Models with Shrinkage*. R package version 2.0.1, URL https://CRAN.R-project.org/package=shrinkTVP.

Markowitz H (1952). "Portfolio Selection." *The Journal of Finance*, **7**(1), 77–91. doi:10.1111/j.1540-6261.1952.tb01525.x.

McCausland WJ, Miller S, Pelletier D (2011). "Simulation Smoothing for State-Space Models: A Computational Efficiency Analysis." *Computational Statistics and Data Analysis*, **55**(1), 199–212. doi:10.1016/j.csda.2010.07.009.

McElreath R (2015). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Texts in Statistical Science, 1st edition. Chapman and Hall, CRC Press. ISBN 9781482253443.

Nakajima J (2012). "Bayesian Analysis of Generalized Autoregressive Conditional Heteroskedasticity and Stochastic Volatility: Modeling Leverage, Jumps and Heavy-Tails for Financial Time Series." *Japanese Economic Review*, **63**(1), 81–103. doi:10.1111/j.1468-5876.2011.00537.x.

Nakajima J, Omori Y (2009). "Leverage, Heavy-Tails and Correlated Jumps in Stochastic Volatility Models." *Computational Statistics and Data Analysis*, **53**(6), 2335–2353. doi:10.1016/j.csda.2008.03.015.

Nakajima J, Omori Y (2012). "Stochastic Volatility Model with Leverage and Asymmetrically Heavy-Tailed Error Using GH Skew Student's t Distribution." *Computational Statistics & Data Analysis*, **56**(11), 3690–3704. doi:10.1016/j.csda.2010.07.012.

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

Omori Y, Chib S, Shephard N, Nakajima J (2007). "Stochastic Volatility with Leverage: Fast and Efficient Likelihood Inference." *Journal of Econometrics*, **140**(2), 425–449. doi:10.1016/j.jeconom.2006.07.008.

Park T, Casella G (2008). "The Bayesian Lasso." *Journal of the American Statistical Association*, **103**(452), 681–686. doi:10.1198/016214508000000337.

Pitt MK, Shephard N (1999). "Time-Varying Covariances: A Factor Stochastic Volatility Approach." In JM Bernardo, JO Berger, AP Dawid, AFM Smith (eds.), *Bayesian Statistics 6 – Proceedings of the Sixth Valencia International Meeting*, pp. 547–570. Oxford University Press.

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/Rnews_2006-1.pdf.

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 J (2007). "Coupling and Ergodicity of Adaptive Markov Chain Monte Carlo Algorithms." *Journal of Applied Probability*, **44**(2), 458–475. doi:10.1239/jap/1183667414.

Rue H (2001). "Fast Sampling of Gaussian Markov Random Fields." *Journal of the Royal Statistical Society B*, **63**(2), 325–338. doi:10.1111/1467-9868.00288.

Sanderson C, Curtin R (2016). "**Armadillo**: A Template-Based C++ Library for Linear Algebra." *Journal of Open Source Software*, **1**, 26–18. doi:10.21105/joss.00026.

Schwalb B, Tresch A, Torkler P, Duemcke S, Demel C, Ripley B, Venables B (2018). ***LSD**: Lots of Superior Depictions*. R package version 4.0-0, URL https://CRAN.R-project.org/package=LSD.

Sentana E, Fiorentini G (2001). "Identification, Estimation and Testing of Conditionally Heteroskedastic Factor Models." *Journal of Econometrics*, **102**(2), 143–164. doi:10.1016/S0304-4076(01)00051-3.

Silva RS, Lopes HF, Migon HS (2006). "The Extended Generalized Inverse Gaussian Distribution for Log-Linear and Stochastic Volatility Models." *Brazilian Journal of Probability and Statistics*, **20**(1), 67–91. URL https://www.jstor.org/stable/43601074.

Sisson SA, Fan Y, Beaumont MA (eds.) (2018). *Handbook of Approximate Bayesian Computation*. Handbooks of Modern Statistical Methods, 1st edition. Chapman and Hall, CRC Press. ISBN 9781439881507.

Taylor SJ (1982). "Financial Returns Modeled by the Product of Two Stochastic Processes: A Study of Daily Sugar Prices 1961–75." In OD Anderson (ed.), *Time Series Analysis, Theory and Practice*, pp. 203–226. North-Holland, Amsterdam.

Wahl JC (2020). ***stochvolTMB**: Likelihood Estimation of Stochastic Volatility Models*. R package version 0.1.2, URL https://CRAN.R-project.org/package=stochvolTMB.

Wickham H, Hester J, Chang W (2020). ***devtools**: Tools to Make Developing R Packages Easier*. R package version 2.3.2, URL https://CRAN.R-project.org/package=devtools.

Yu Y, Meng XL (2011). "To Center or not to Center: That is not the Question—An Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency." *Journal of Computational and Graphical Statistics*, **20**(3), 531–570. 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.

Zhou X, Nakajima J, West M (2014). "Bayesian Forecasting and Portfolio Decisions Using Dynamic Dependent Sparse Factor Models." *International Journal of Forecasting*, **30**(4), 963–980. doi:10.1016/j.ijforecast.2014.03.017.

Affiliation:

Darjus Hosszejni
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
WU Vienna University of Economics and Business
Welthandelsplatz 1 / Building D4 / Level 4
1020 Vienna, Austria

E-mail: Darjus.Hosszejni@wu.ac.at
URL: https://statmath.wu.ac.at/~hosszejni/

Gregor Kastner
Department of Statistics
University of Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt, Austria
E-mail: Gregor.Kastner@aau.at
URL: https://www.aau.at/team/gregor-kastner/