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

Darjus Hosszejni

WU Vienna University
of Economics and Business

Gregor Kastner

WU Vienna University
of Economics and Business

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 four 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, heavy-tailed SV, and SV with leverage. 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.

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; Gysels, 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 2019) 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** (Kastner and Hosszejni 2019) with several practically relevant univariate methods. Second, we introduce the new companion package **factorstochvol** (Kastner 2019a) which focuses on the multivariate case. The extended **stochvol** now provides the means for the Bayesian estimation of vanilla SV, heavy-tailed SV and 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 two relaxed models: the SV model with Student's t errors (SVt) and the SV model with leverage (SVl). Finally, we close the section after discussing prior distributions and 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 $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ denote a vector of observations. The SV model assumes the following structure for $\boldsymbol{y}$,

$$
\begin{aligned}
y_t \mid \boldsymbol{x}_t, \boldsymbol{\beta}, h_t &\sim \mathcal{N}(\boldsymbol{x}_t \boldsymbol{\beta}, \exp(h_t)), \\
h_{t+1} \mid \boldsymbol{\vartheta}, h_t &\sim \mathcal{N}(\mu + \varphi(h_t - \mu), \sigma^2),
\end{aligned}
\tag{1}
$$

where $\mathcal{N}(a, b)$ denotes the normal distribution with mean a and variance b. $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top)^\top$ 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 $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)^\top$, and $\boldsymbol{h} = (h_1, \ldots, h_n)^\top$ denotes the

log-variance process. 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). In the following, we describe the SV model with Student's t errors implemented in **stochvol**.

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

where $t_\nu(a, b)$ is the Student's t distribution with ν degrees of freedom, mean a, and scale b. The single difference between Equation 1 and Equation 2 is that the observations are conditionally t distributed. Hence, Equation 2 relaxes 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,

$$
\begin{aligned}
\begin{pmatrix} y_t \\ h_{t+1} \end{pmatrix} \Bigg| h_t, \boldsymbol{\zeta}, \boldsymbol{x}_t, \boldsymbol{\beta} &\sim \mathcal{N}_2\left(\begin{pmatrix} \boldsymbol{x}_t\boldsymbol{\beta} \\ \mu + \varphi(h_t - \mu) \end{pmatrix}, \boldsymbol{\Sigma}^\rho\right), \\
\boldsymbol{\Sigma}^\rho &= \begin{pmatrix} \exp(h_t) & \rho\sigma\exp(h_t/2) \\ \rho\sigma\exp(h_t/2) & \sigma^2 \end{pmatrix},
\end{aligned}
\tag{3}
$$

where $\mathcal{N}_2(\boldsymbol{b}, \boldsymbol{B})$ is the two-dimensional normal distribution with mean vector $\boldsymbol{b} \in \mathbb{R}^2$ and covariance matrix $\boldsymbol{B} \in \mathbb{R}^{2\times 2}$. 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 $\rho = 0$.

2.2. Prior distributions

We a priori assume $\beta_k \sim \mathcal{N}(b_\beta, B_\beta)$ independently for $k = 1, \ldots, K$. By setting $b_\beta = 0$, 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). It allows strong prior information and thus shrinkage toward b_β for small values of B_β, but it can also serve as a rather uninformative prior for large values of B_β.

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 $a_\varphi \approx 1.5$ and $b_\varphi \gtrsim 5$.

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. The commonly applied and convenient conjugate prior $\sigma^{-2} \sim \mathcal{G}(a_\sigma, b_\sigma)$, where $\mathcal{G}(a_\sigma, b_\sigma)$ is the gamma distribution with shape parameter a_σ and rate parameter b_σ, bounds σ away from 0 and it is therefore in our view an unsatisfactory choice. Following Frühwirth-Schnatter and Wagner (2010) and Kastner and Frühwirth-Schnatter (2014), we advocate and employ $\pm\sigma \sim \mathcal{N}(0, B_\sigma)$ instead, which corresponds to $\sigma^2 \sim \mathcal{G}(1/2, 1/(2B_\sigma))$.

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.

Equation 2 – the SV model with Student's t errors – additionally requires 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 $\nu > 2$. Moreover, when ν is large and consequently the likelihood is insensitive to variations in ν, bounding ν from above can alleviate mixing problems. Consequently, we equip ν with a uniform prior $\nu \sim \mathcal{U}(a_\nu, b_\nu)$, where we require $a_\nu \geq 2$.

Finally, in the case of the SV model 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

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. 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 Markov chain Monte Carlo (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

m	free elements of $\boldsymbol{\Sigma}_t$	free elements of $\boldsymbol{\Sigma}_t$ 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 $\boldsymbol{\Sigma}_t$ for different numbers of component series m.

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, and SVl 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, and 3 similar to the ones in Kim *et al.* (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.

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). 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

m	free elements of $\boldsymbol{\Sigma}_t$	free elements of $\boldsymbol{\Sigma}_t$ 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 $\boldsymbol{\Sigma}_t$ in a factor model for different numbers of component series m and number of factors $r = 4$.

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 decomposition

$$\boldsymbol{\Sigma}_t = \check{\boldsymbol{\Sigma}}_t + \bar{\boldsymbol{\Sigma}}_t, \tag{4}$$

where $\mathrm{rank}(\check{\boldsymbol{\Sigma}}_t) = r < m$, and $\bar{\boldsymbol{\Sigma}}_t$ is the diagonal matrix containing the variances of the idiosyncratic errors. The rank assumption on the symmetric $\check{\boldsymbol{\Sigma}}_t$ gives rise to the factorization $\check{\boldsymbol{\Sigma}}_t = \Psi\Psi^\top$, 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 $\boldsymbol{\Sigma}_t$, 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.

$$
\begin{aligned}
\boldsymbol{y}_t \mid \boldsymbol{\Lambda}, \boldsymbol{f}_t, \bar{\boldsymbol{\Sigma}}_t &\sim \mathcal{N}_m(\boldsymbol{\Lambda}\boldsymbol{f}_t, \bar{\boldsymbol{\Sigma}}_t), \\
\boldsymbol{f}_t \mid \tilde{\boldsymbol{\Sigma}}_t &\sim \mathcal{N}_r(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}}_t),
\end{aligned}
\tag{5}
$$

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

$$
\begin{aligned}
\bar{\boldsymbol{\Sigma}}_t &= \mathrm{diag}(\exp(\bar{h}_{t1}), \ldots, \exp(\bar{h}_{tm})), \\
\tilde{\boldsymbol{\Sigma}}_t &= \mathrm{diag}(\exp(\tilde{h}_{t1}), \ldots, \exp(\tilde{h}_{tr}))), \\
\bar{h}_{ti} &\sim \mathcal{N}(\bar{\mu}_i + \bar{\varphi}_i(\bar{h}_{t-1,i} - \bar{\mu}_i), \bar{\sigma}_i^2), \quad i = 1, \ldots, m, \\
\tilde{h}_{tj} &\sim \mathcal{N}(\tilde{\mu}_j + \tilde{\varphi}_j(\tilde{h}_{t-1,j} - \tilde{\mu}_j), \tilde{\sigma}_j^2), \quad j = 1, \ldots, r.
\end{aligned}
\tag{6}
$$

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 5, we can reformulate Equation 4 as

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Lambda}\tilde{\boldsymbol{\Sigma}}_t\boldsymbol{\Lambda}^\top + \bar{\boldsymbol{\Sigma}}_t, \tag{7}$$

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[1] $\boldsymbol{P}$ of

---

[1]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.

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 $\boldsymbol{\Lambda}' = \boldsymbol{\Lambda} \boldsymbol{P}^\top$ 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 latent log-variance processes as well as for the factor loadings matrix $\boldsymbol{\Lambda}$. The former 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,

$$\Lambda_{ij} \mid \tau_{ij}^2 \sim \mathcal{N}(0, \tau_{ij}^2), \quad \tau_{ij}^2 \mid \lambda_i^2 \sim \mathcal{G}(a, a\lambda_i^2/2). \tag{8}$$

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 8 induces row-wise shrinkage with element-wise adaption. Analogously, one could also consider column-wise shrinkage with element-wise adaption, i.e.,

$$\Lambda_{ij} \mid \tau_{ij}^2 \sim \mathcal{N}(0, \tau_{ij}^2), \quad \tau_{ij}^2 \mid \lambda_j^2 \sim \mathcal{G}(a, a\lambda_j^2/2), \tag{9}$$

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 bad MCMC mixing 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 Kastner *et al.* (2017, using Gaussian priors for the factor loadings) as well as Kastner (2019b, 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_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, and SVl models via its sampling routines `svsample()`, `svtsample()`, and `svlsample()`, 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, 2008 and March 1, 2012 (1028 data points) in three different ways.

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> library("stochvol")
R> data("exrates")
R> ind <- which(exrates$date >= as.Date("2008-03-01") &
+    exrates$date <= as.Date("2012-03-01"))
R> CHF_price <- exrates$CHF[ind]
R> res_sv <- svsample(CHF_price, designmatrix = "ar1")
```

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> 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.

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

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)
```

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 development 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.
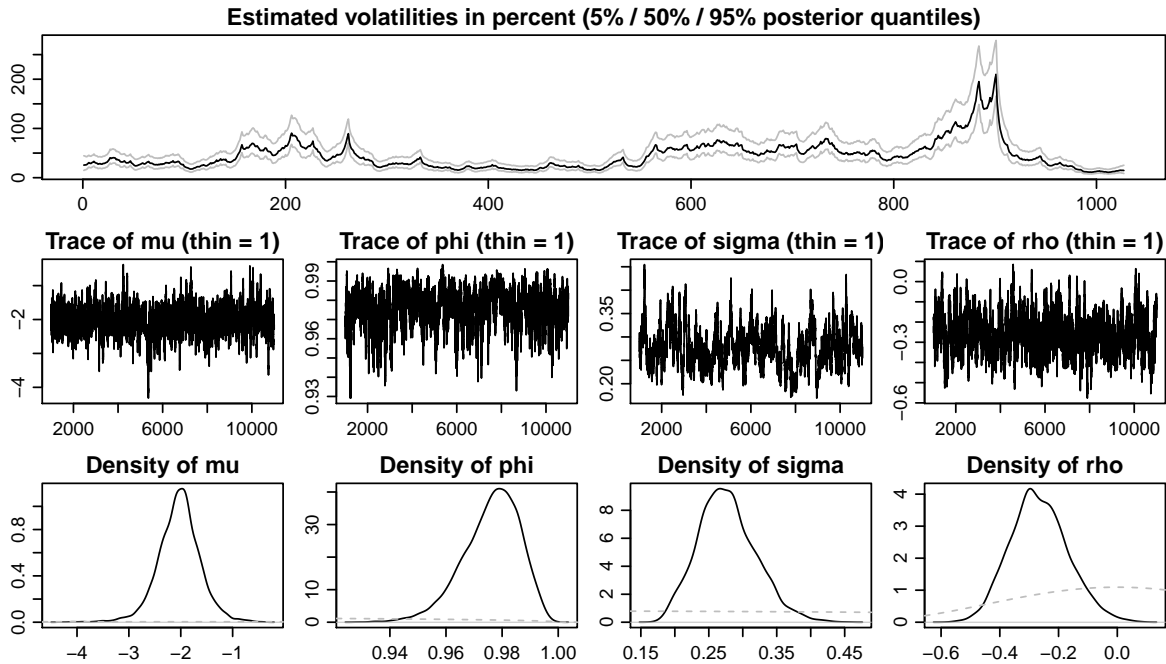
Next, we observe the AR coefficients.

Figure 1: The default plot of an estimated model. The top row shows a summary of the posterior of the 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.

```
R> for (i in seq_len(3)) {
+     traceplot(res_svl$beta[, i])
+     densplot(res_svl$beta[, 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 10000 MCMC draws after a burn-in of 1000.
Prior distributions:
mu        ~ Normal(mean = 0, sd = 100)
(phi+1)/2 ~ Beta(a0 = 5, b0 = 1.5)
sigma^2   ~ 1 * Chisq(df = 1)
(rho+1)/2 ~ Beta(a0 = 4, b0 = 4)
```
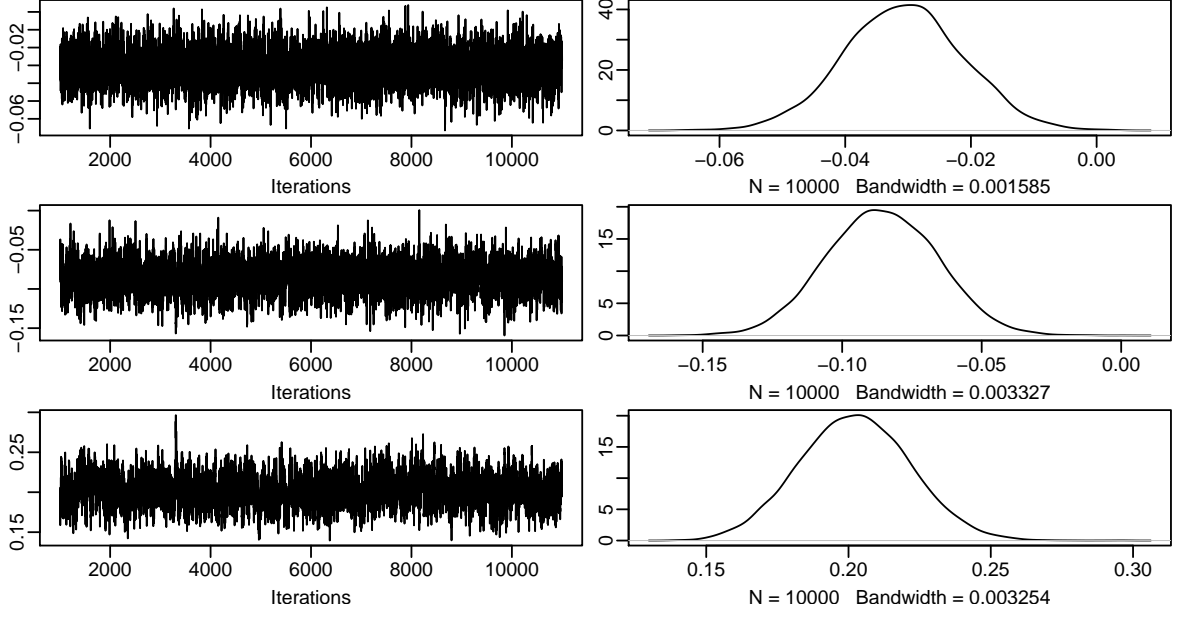
Figure 2: Trace plots and estimated kernel densities of posterior draws from $p(\boldsymbol{\beta} \mid \boldsymbol{y})$.

```
Posterior draws of parameters (thinning = 1):
            mean     sd     5%     50%     95%  ESS
mu        -2.033  0.410 -2.687 -2.023 -1.384  509
phi        0.976  0.010  0.958  0.977  0.991  207
sigma      0.277  0.043  0.211  0.274  0.350   85
rho       -0.266  0.098 -0.420 -0.271 -0.096  241
exp(mu/2)  0.369  0.076  0.261  0.364  0.501  509
sigma^2    0.079  0.024  0.044  0.075  0.122   85
```

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. 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()` 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. 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

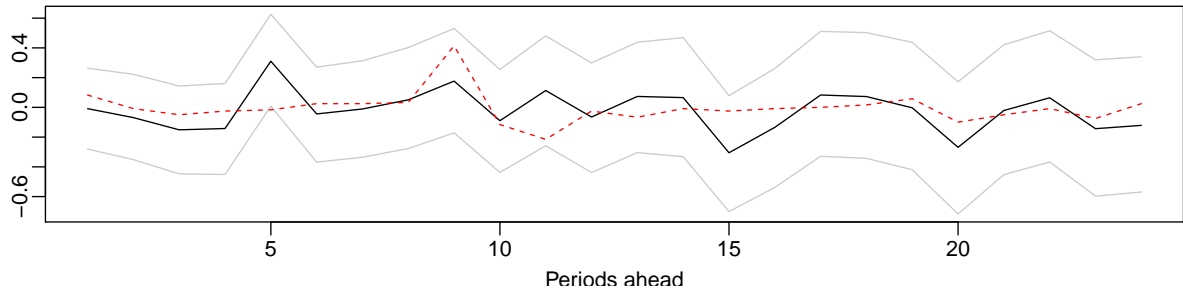We employ our estimated model to predict log returns for the remaining days in the data set.

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

To do so, we first prepare the covariates for the next 24 days and pass them via the argument `newdata` of the generic `predict()` function along with the estimation output. Note that we need 25 days of price data to obtain 24 returns.

```
R> pred_ind <- seq(tail(ind, 1), length.out = 25)
R> pred_X <- cbind(constant = 1,
+    100 * logret(exrates$USD[pred_ind]),
+    100 * logret(exrates$JPY[pred_ind]))
R> pred_svl <- predict(res_svl, 24, 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 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$CHF[pred_ind])
R> ts.plot(cbind(t(apply(pred_svl$y, 2, quantile, c(0.05, 0.5, 0.95))),
+    obs_CHF), xlab = "Periods ahead", lty = c(rep(1, 3), 2),
+    col = c("gray80", "black", "gray80", "red"))
```

4.4. 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()`, and `svlsample()` 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 the prior of regression coefficients by providing $(b_\beta, \sqrt{B_\beta})$. The prior of ν can be influenced in `svtsample()` by passing (a_ν, b_ν) as the argument `priornu`. Finally, `svlsample()` takes 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, priornu = c(2, 50), priorbeta = c(0, 10000))
R> svlsample(CHF_logret, priormu = c(0, 100), priorphi = c(5, 1.5),
+     priorsigma = 1, priorrho = c(4, 4), priorbeta = c(0, 10000))
```

4.5. 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, which in our experience is enough for most applications. As for the initial values, we find them to be of minor importance as the samplers tend to mix well enough in real world data sets. Nevertheless, `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 $\boldsymbol{h}$. 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 observations of 23 EUR exchange rates, ranging from March 3, 2000 to April 4, 2012. To keep the analysis simple and computation times moderate, we however only model the last 1001 days of the first six series (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 which we demean component-wise. This leaves us with a data set of size $n = 1000$ and $m = 6$. The data is prepared using the code snippet below and visualized in Figure 4.

```
R> library("factorstochvol")
R> data("exrates", package = "stochvol")
```
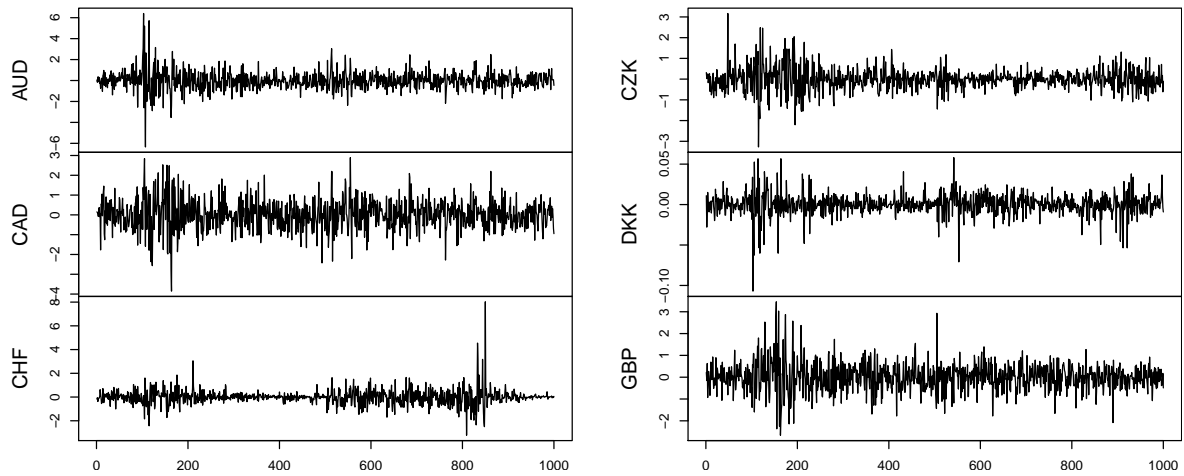
Figure 4: Percentage log returns of six EUR exchange rates.

```
R> m <- 6
R> n <- 1000
R> y <- 100 * logret(tail(exrates[, seq_len(m)], n + 1), demean = TRUE)
R> plot.ts(y)
```

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()`. 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)
```

```
[1] 2 3 1 4 5 6
```

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. Below is the result for the data set at hand.

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


      [,1]  [,2]
[1,] FALSE FALSE
[2,] FALSE  TRUE
[3,] FALSE FALSE
[4,] FALSE FALSE
[5,] FALSE FALSE
[6,] FALSE FALSE
```

If `fsvsample()` is called with the default 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 7. 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 2019b).

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()` and `svlsample()`. In the following, $i = 1, \ldots, m$ and $j = 1, \ldots, r$ index the idiosyncratic and the factor log-variance processes, respectively. 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 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. For a complete list of all arguments and more details, see ?fsvsample.

- y: the data;

- factors: the number of factors;

- draws: the number of MCMC samples to be drawn after burnin;

- thin: the amount of thinning (every thin[th] draw is kept);

- burnin: 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,

- keeptime: 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: indicator(s) to turn off stochastic volatility for the idiosyncratic variances, the factor variances, or both;

- samplefac: indicator to turn off sampling of the factors; useful to work with observed instead of latent factors (see Kastner 2019b, for a use case of this);

- runningstore: to avoid having to store all MCMC draws, fsvsample's default is to compute and store the first runningstoremoments (two by default) ergodic moments of some interesting quantities (such as 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;

- runningstorethin: 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: a logical indicator determining the verbosity of fsvsample.

For our illustrative example, most settings are left at their default values. Only the number of factors is increased from one to two, instead of 1000 we sample 10000 draws, a thinning of 10 is used, and `quiet` is set to `TRUE`.

```
R> library(factorstochvol)
R> res <- fsvsample(y, factors = 2, draws = 10000, thin = 10, quiet = TRUE)
```

5.5. Investigating the output and visualizing the results

The resulting object

```
R> res
```

```
Fitted factor stochastic volatility object with
    -       6 series
    -       2 factor(s)
    -    1000 timepoints
    -   10000 MCMC draws
    -      10 thinning
    -    1000 burn-in
```

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

- 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 $\bar{\boldsymbol{h}}$,

- 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 quantitites 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))
```

```
[1]    6    6 1000    1
```

shows that we have stored 1000 posterior draws of a 6×6 covariance matrix at one point in time, $t = n = 1000$. To check convergence, one can take a look at the trace plot and the autocorrelation function of the log determinant, i.e.:
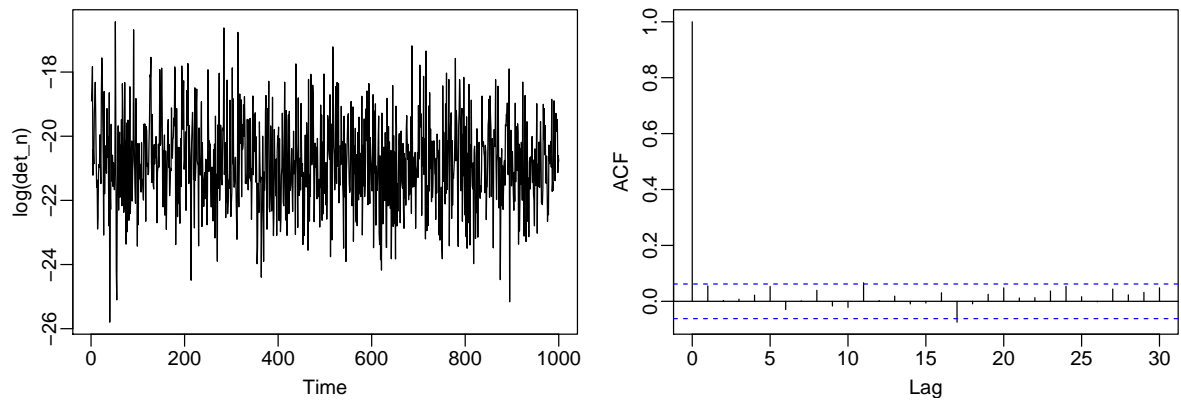
Figure 5: Trace plot and empirical autocorrelation function of the log determinant of the model-implied covariance matrix at $t = n$.

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

The result 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 1000 draws kept) which is implemented in **coda**. Again, no major convergence problems are apparent.

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

```
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1000 1000  899  744 1000 1000
[2,] 1000 1000  566 1069 1000  817
[3,]  899  566  825  512  335  342
[4,]  744 1069  512  864  613  837
[5,] 1000 1000  335  613 1097 1000
[6,] 1000  817  342  837 1000 1000
```

Assuming that `runningstore` was set sufficiently high when sampling, several convenience functions can be used for quick visualizations without having to a posteriori collect 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,
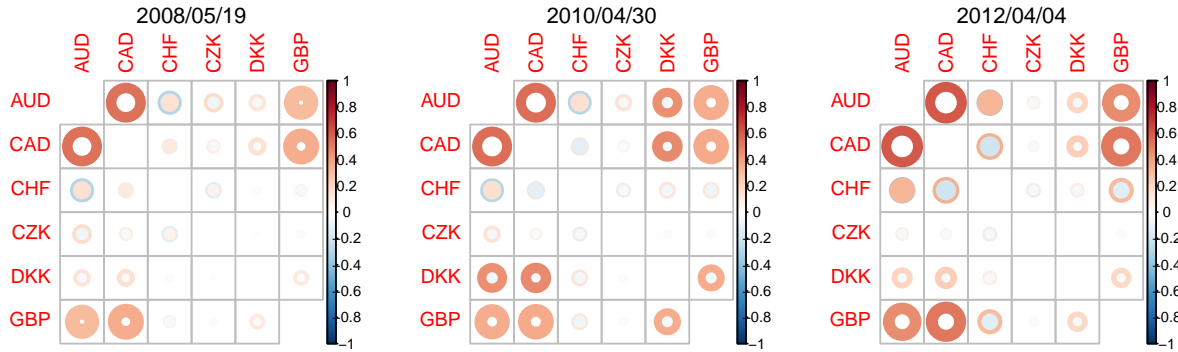
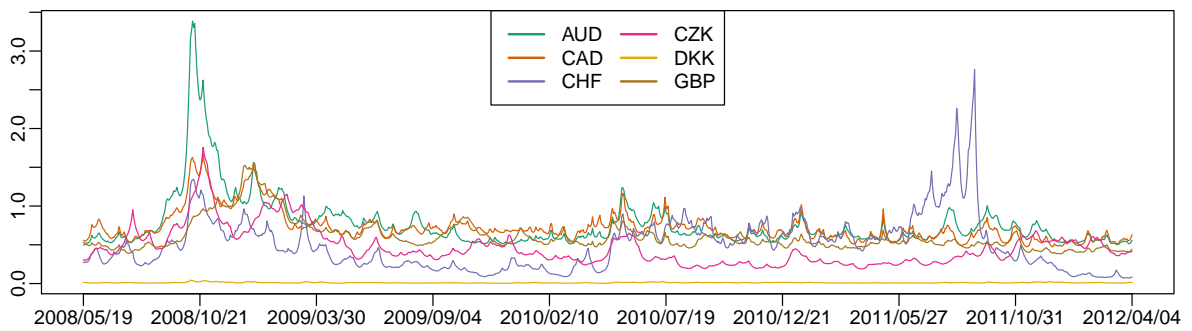Figure 6: Three estimated correlation matrices.

Figure 7: Posterior means of marginal volatilities.

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

yields the estimated volatilities in Figure 7. The financial cirisis 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, from a EUR point of view,
the estimated correlation between AUD and CAD appears to be relatively stable over time,
correlations with 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 examin-
ing the latent factors and their variances themselves. To visualize the loadings, the functions,
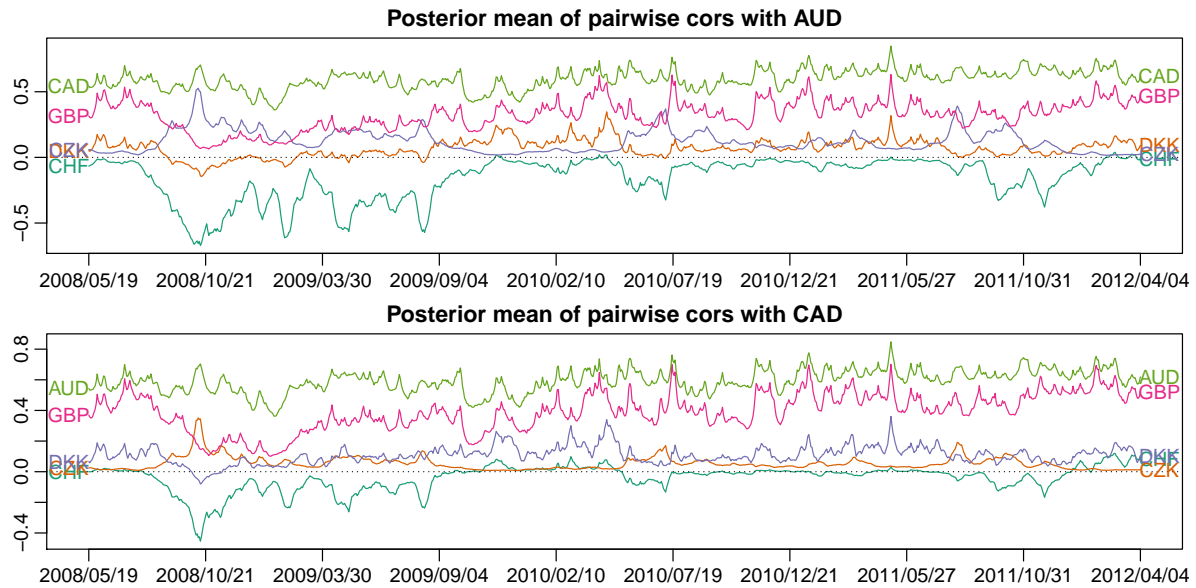`facloadpairplot()`, `facloadcredplot()`, `facloadpointplot()`, `facloadtraceplot()`, and

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

`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")`. Similarily, `logvartimeplot(res, show = "idi")` produces plots of the idiosyncratic log variances which are displayed in Figure 11.

Finally, 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.

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. 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> predcor1 <- predcor(res)
R> round(apply(predcor1[,,,1], 1:2, mean), 2)

      AUD  CAD   CHF   CZK  DKK  GBP
AUD  1.00 0.61 -0.02  0.03 0.09 0.46
CAD  0.61 1.00  0.08  0.01 0.11 0.51
CHF -0.02 0.08  1.00 -0.03 0.03 0.09
CZK  0.03 0.01 -0.03  1.00 0.00 0.01
DKK  0.09 0.11  0.03  0.00 1.00 0.09
GBP  0.46 0.51  0.09  0.01 0.09 1.00

R> round(apply(predcor1[,,,1], 1:2, sd), 2)
```
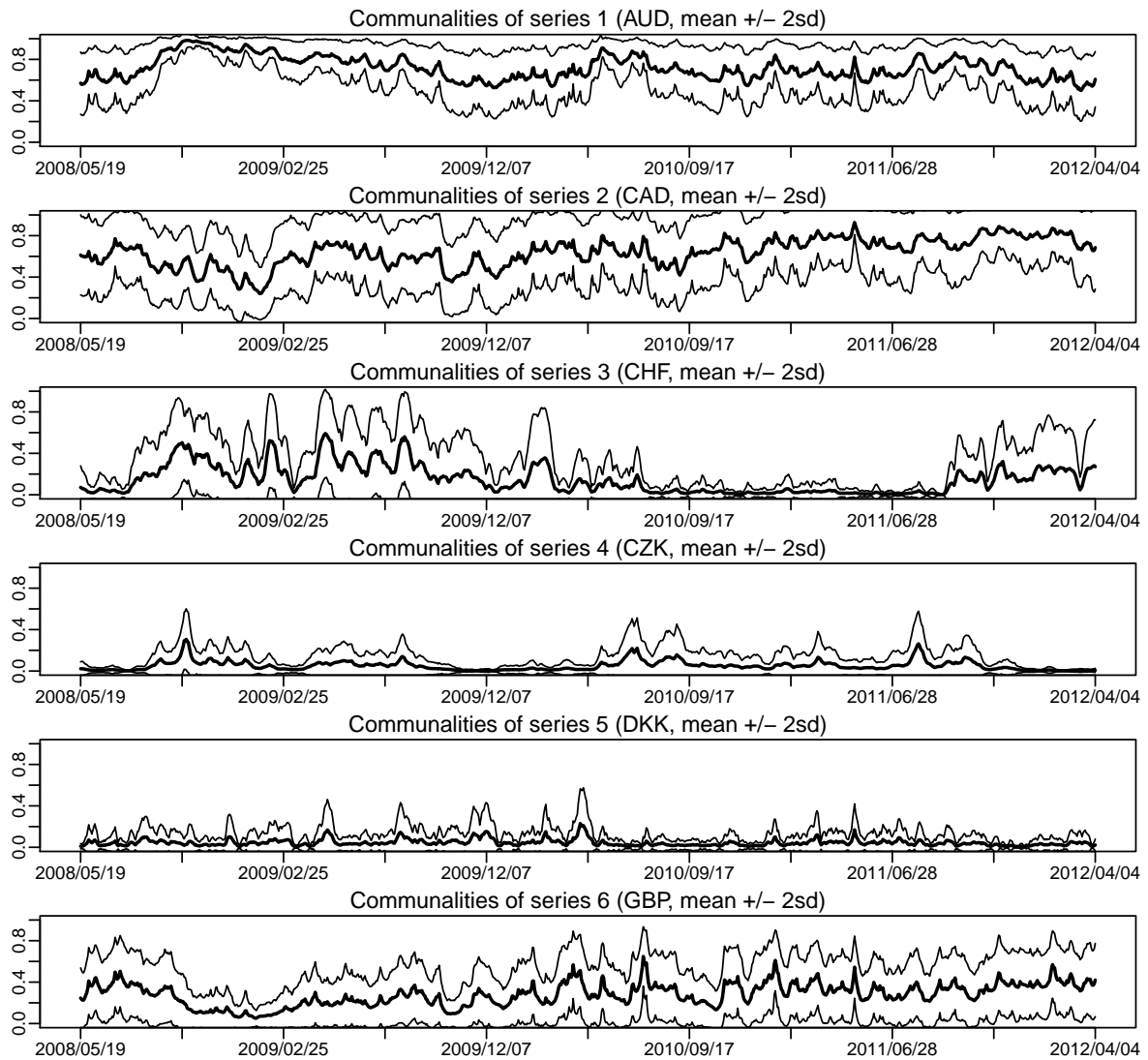
Figure 9: Communalities: Posterior means plus/minus two posterior standard deviations.

```
      AUD  CAD  CHF  CZK  DKK  GBP
AUD 0.00 0.15 0.18 0.04 0.07 0.16
CAD 0.15 0.00 0.14 0.02 0.07 0.17
CHF 0.18 0.14 0.00 0.04 0.03 0.11
CZK 0.04 0.02 0.04 0.00 0.01 0.02
DKK 0.07 0.07 0.03 0.01 0.00 0.06
GBP 0.16 0.17 0.11 0.02 0.06 0.00
```

To obtain draws from the posterior predictive distribution of new data points, one can simply draw from the corresponding scale mixture of multivariate normals. In Figure 12, these draws are visualized via `heatpairs()` from **LSD** (Schwalb, Tresch, Torkler, Duemcke, Demel, Ripley, and Venables 2018).

```
R> predcov_1 <- predcov(res)
```
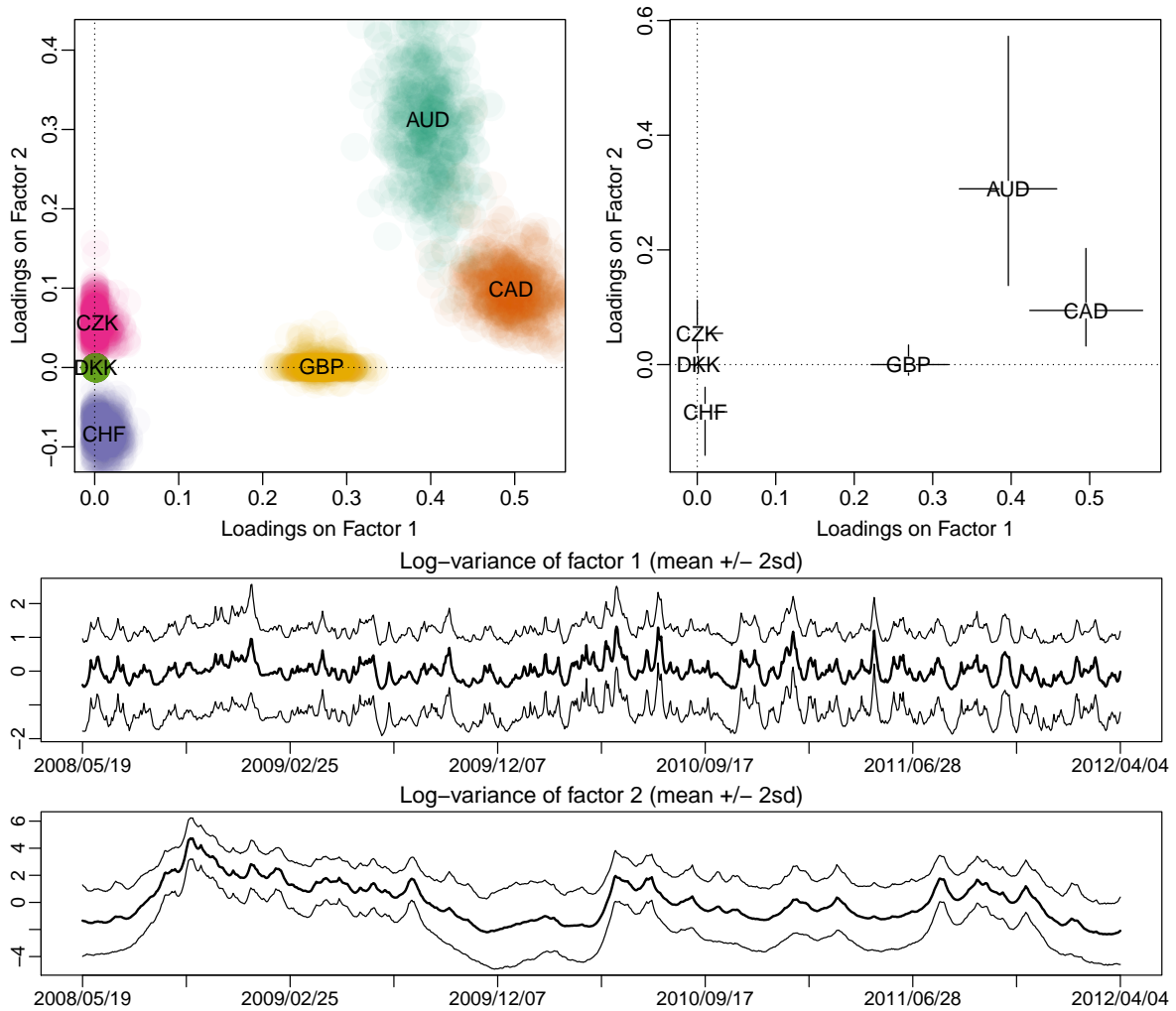
Figure 10: Posterior distribution of the factor loadings (top) and posterior means plus/minus two standard deviations of factor log variances (bottom).

```
R> effectivedraws <- res$config$draws/res$config$thin
R> preddraws <- matrix(NA_real_, effectivedraws, m)
R> for (i in seq_len(effectivedraws))
+    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.
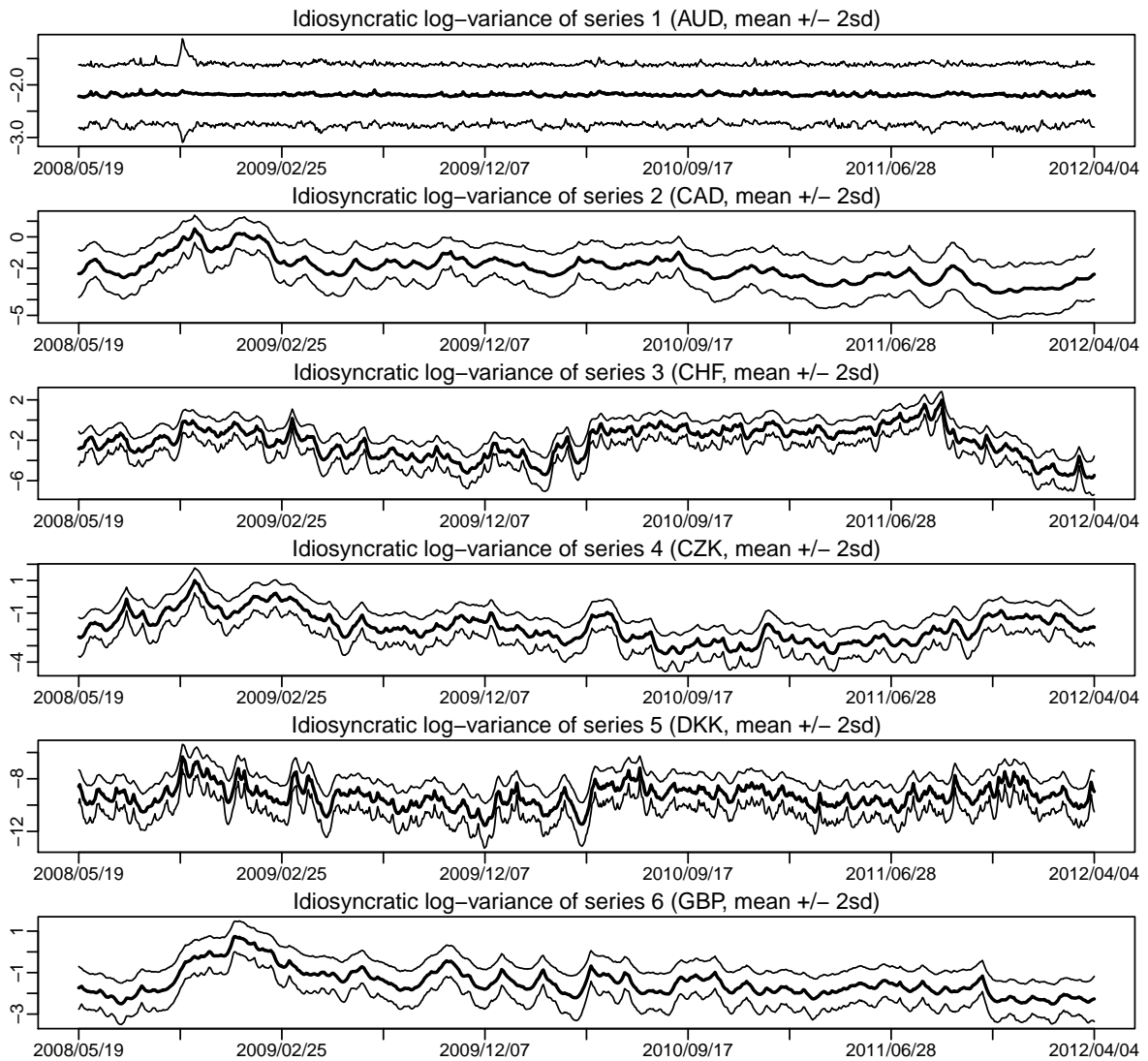
Figure 11: Idiosyncratic log variances: Posterior means plus/minus two standard deviations.

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

       1        2
5.126926 5.093654
```

6. Summary and discussion

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
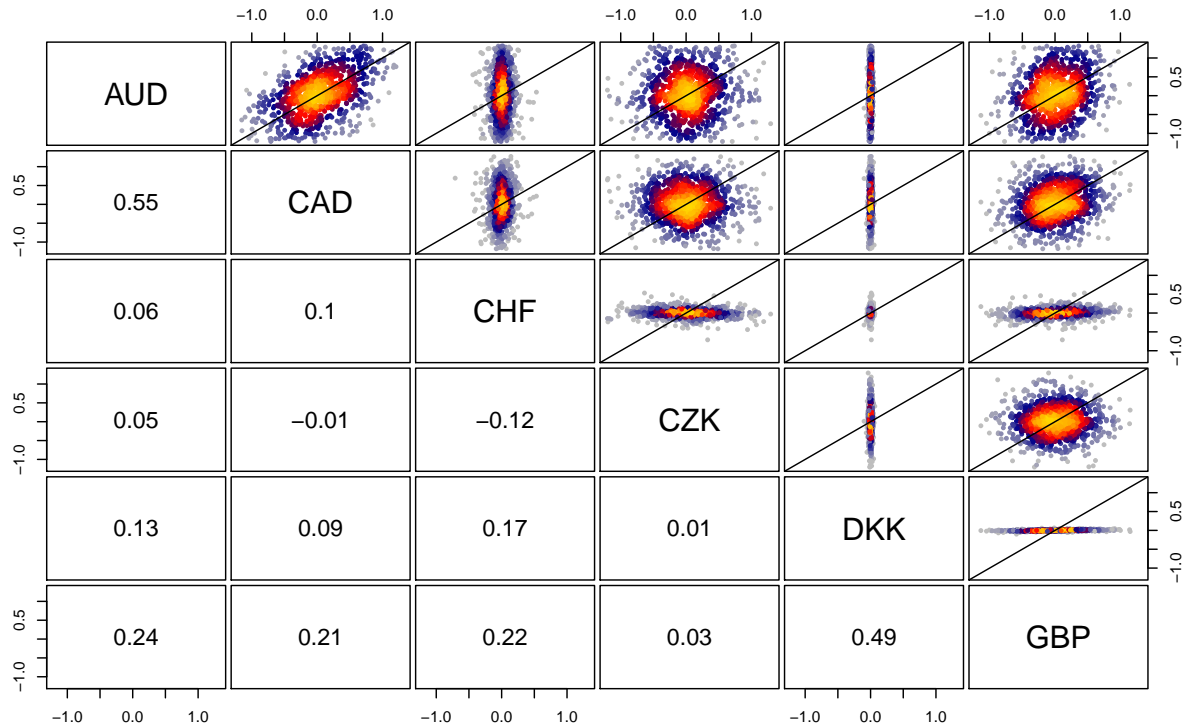
Figure 12: Predictive draws from the one-step-ahead posterior predictive distribution (above the diagonal) and empirical correlation coefficients (below the diagonal).

`update_sv()` and `update_svl()` 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 2019).

References

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.

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, 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.

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.

Gysels 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.

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*, Springer Proceedings in Mathematics & Statistics. Springer-Verlag. Forthcoming, URL https://arxiv.org/abs/1901.11491.

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**." *R package vignette*, Comprehensive R Archive Network (CRAN). URL https://CRAN.R-project.org/package=stochvol/vignettes/heavytails.pdf.

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 (2019a). **factorstochvol**: *Bayesian Estimation of (Sparse) Latent Factor Stochastic Volatility Models.* R package version 0.9.2, URL https://cran.r-project.org/package=factorstochvol.

Kastner G (2019b). "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 (2019). **stochvol***: Efficient Bayesian Inference for Stochastic Volatility (SV) Models.* R package version 2.0.4, URL http://CRAN.R-project.org/package=stochvol.

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 (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.

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 (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 (2019). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. 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.

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.

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, Gregor Kastner
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, gregor.kastner@wu.ac.at
URL: http://statmath.wu.ac.at/~hosszejni/, http://statmath.wu.ac.at/~kastner/