Dealing with Stochastic Volatility in Time Series Using the R Package stochvol

Gregor Kastner
WU Vienna University of Economics and Business

Abstract

Heteroskedasticity in financial and economic datasets is a commonly observed feature, and the need to model this feature properly has for a long time been of great interest to researchers and practitioners alike. The R package **stochvol** provides a fully Bayesian implementation of heteroskedasticity modelling by means of the *stochastic volatility* framework. Being a state-space formulation, it describes contemporaneous volatilities as latent random variables as opposed to deterministic values. The software described in this paper utilizes Markov chain Monte Carlo (MCMC) samplers to conduct inference by obtaining draws from the posterior distribution of parameters and latent variables, which can then be used for predicting future volatilities. The package can straightforwardly be employed as a stand-alone tool; moreover, it allows for easy incorporation into other MCMC samplers. Main focus of the paper is to show the functionality of **stochvol**, nevertheless it also provides a brief mathematical description of the model, an overview of the sampling schemes used, and an in-depth example using exchange rate data.

Keywords: Bayesian inference, Markov chain Monte Carlo (MCMC), auxiliary mixture sampling, ancillarity-sufficiency interweaving strategy (ASIS), state-space model, heteroskedasticity, financial time series.

1. Introduction

When analyzing (financial) returns, focus is often laid on estimating and predicting potentially time varying volatilities. This interest has a long history, dating at least back to Markowitz (1952), who investigated portfolio construction with optimal expected return-variance trade-off. In his article, he proposes rolling-window-type estimates for the instantaneous volatilities, but already then recognizes the potential for "better methods, which take into account more information".

One way of doing so is to model the evolution of volatility deterministically, i.e., through the (G)ARCH class of models. After the groundbreaking papers of Engle (1982) and Bollerslev (1986), these models have been generalized in numerous ways and were applied to a vast amount of real-world problems. As an alternative, Taylor (1982) proposes in his seminal work to model the volatility probabilistically, i.e., through a state-space model where the logarithm of the squared volatilities – the latent states – follow an autoregressive process of order one. Over time, this specification became known as the *stochastic volatility (SV)* model. Even though several papers (e.g. Jacquier, Polson, and Rossi 1994; Ghysels, Harvey, and Renault 1996; Kim, Shephard, and Chib 1998) find early evidence in favor of using SV, these models

have found comparably little use in applied work. Reviewing this discrepancy, Bos (2012) states:

> While there are literally thousands of applications of GARCH, for SV, this number is far lower. Two reasons for this relative lack of applied work using SV are apparent. First, there are as of yet no standard packages for estimating SV models, whereas for GARCH, most statistical packages have a wealth of options for incorporating GARCH effects. A second difference seems to be that GARCH has many variants of the model (Bollerslev 2008), with basically a single estimation method for all of them. For SV, there are few variants of the model, but a full series of estimation methods.

In Kastner and Frühwirth-Schnatter (2014), the latter issue has been thoroughly investigated, and an efficient MCMC estimation scheme has been proposed. The paper at hand and the corresponding package **stochvol** (Kastner 2014) for R (R Core Team 2014) were crafted to cope with the first problem: the apparent lack of standard packages for efficiently estimating SV models.

2. Model specification and estimation

We briefly introduce the model and specify the notation used in the remainder of the paper. Furthermore, a quick overview over Bayesian parameter estimation via Markov chain Monte Carlo (MCMC) methods is given.

2.1. The SV model

Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)^\top$ be a vector of returns with mean zero. The intrinsic feature of the SV model is that each observation y_t is assumed to have its "own" contemporaneous variance e^{h_t}, thus relaxing the usual assumption of homoskedasticity. In order to make estimation of such a model feasible (note that there are just as many data points as there are variances!), this variance is not allowed to vary unrestrictedly with time. Rather, its logarithm is assumed to follow a parametric stochastic process, more specifically: an autoregressive process of order one. Note that feature is fundamentally different to GARCH-type models, where the time-varying volatility is assumed to follow a deterministic rather than stochastic evolution.

The SV model (in its centered parameterization) is thus given through

$$y_t|h_t \sim \mathcal{N}(0, \exp h_t)\,, \tag{1}$$

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

$$h_0|\mu, \phi, \sigma_\eta \sim \mathcal{N}\big(\mu, {\sigma_\eta}^2/(1-\phi^2)\big)\,, \tag{3}$$

where $\mathcal{N}\big(\mu, {\sigma_\eta}^2\big)$ denotes the normal distribution with mean μ and variance ${\sigma_\eta}^2$. We will refer to $\boldsymbol{\theta} = (\mu, \phi, \sigma_\eta)^\top$ as the vector of *parameters*: the *level* μ, the *persistence* ϕ, and the *volatility* σ_η of log-variance. The process $\boldsymbol{h} = (h_0, h_1, \ldots, h_T)$ appearing in state equations (2) and (3) is unobserved and usually interpreted as the latent time-varying *volatility process* (more precisely: the log-variance process). Note that the initial state h_0 is distributed according to the stationary distribution of the autoregressive process of order one.

2.2. Prior distributions

To complete the model setup, a prior distribution for the parameter vector $\boldsymbol{\theta}$ needs to be specified. Following Kim *et al.* (1998), we choose independent components for each parameter, i.e., $p(\boldsymbol{\theta}) = p(\mu)p(\phi)p(\sigma_\eta)$.

The level $\mu \in \mathbb{R}$ is equipped with the usual normal prior $\mu \sim \mathcal{N}(b_\mu, B_\mu)$. In practical applications, this prior is usually chosen to be rather uninformative, e.g., through setting $b_\mu = 0$ and $B_\mu \geq 100$ for daily log-returns. Our experience with empirical data is that the exact choice is usually not very influential; see also Section 3.2.

For the persistence parameter $\phi \in (-1, 1)$, we choose $(\phi + 1)/2 \sim \mathcal{B}(a_0, b_0)$, implying

$$p(\phi) = \frac{1}{2B(a_0, b_0)} \left(\frac{1+\phi}{2}\right)^{a_0-1} \left(\frac{1-\phi}{2}\right)^{b_0-1}, \tag{4}$$

where a_0 and b_0 are positive hyperparameters and $B(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt$ denotes the beta function. Clearly, the support of this distribution is the unit interval $(-1, 1)$; thus, stationarity of the autoregressive volatility process is guaranteed. Its expected value and variance are given through the expressions

$$\begin{aligned} E(\phi) &= \frac{2a_0}{a_0 + b_0} - 1, \\ V(\phi) &= \frac{4a_0 b_0}{(a_0 + b_0)^2(a_0 + b_0 + 1)}. \end{aligned}$$

This obviously implies that the prior expectation of ϕ depends only on the ratio $a_0 : b_0$. It is greater than zero if and only if $a_0 > b_0$ and smaller than zero if and only if $a_0 < b_0$. For a fixed ratio $a_0 : b_0$, the prior variance decreases with larger values of a_0 and b_0. The uniform distribution on the unit interval arises as a special case when $a_0 = b_0 = 1$. For financial datasets with not too many observations (i.e., $T \lesssim 1000$), the choice of the hyperparameters a_0 and b_0 can be quite influential on the shape of the posterior distribution of ϕ. In fact, note that in the case when the underlying data-generating process is (near-)homoskedastic, the volatility of log-variance σ_η will be (very close to) zero and thus the likelihood will contain little to no information about ϕ. Consequently, the posterior distribution of ϕ will be (almost) equal to its prior, no matter how many data points are observed. For some discussion about this issue, see e.g., Kim *et al.* (1998), who choose $a_0 = 20$ and $b_0 = 1.5$, implying a prior mean of 0.86 with a prior standard deviation of 0.11 and thus very little mass for nonpositive values of ϕ.

For the volatility of log-variance $\sigma_\eta \in \mathbb{R}^+$, we choose ${\sigma_\eta}^2 \sim B_{\sigma_\eta} \times \chi^2_1 = \mathcal{G}\left(1/2, 1/2B_{\sigma_\eta}\right)$, which is motivated by Frühwirth-Schnatter and Wagner (2010), who equivalently stipulate the prior for $\pm\sqrt{{\sigma_\eta}^2}$ to follow a centered normal distribution, i.e., $\pm\sqrt{{\sigma_\eta}^2} \sim \mathcal{N}\left(0, B_{\sigma_\eta}\right)$. As opposed to the usual Inverse-Gamma prior, this prior is not conjugate in the usual sampling scheme, however, does not bound σ_η away from zero a priori. The choice of the hyperparameter B_{σ_η} turns out to be of minor influence in empirical applications, as long as it is not picked to be too small.

2.3. MCMC Sampling

An MCMC algorithm such as the one implemented in the package **stochvol** will provide its user with draws from the posterior distribution of the desired random variables: in our case

the latent log-variances $\boldsymbol{h}$ and the parameter vector $\boldsymbol{\theta}$. Because these draws are usually dependent, Bayesian inference via MCMC may require careful design of the algorithm and attentive investigation of the draws obtained.

One key feature of the algorithm used in this package is the joint sampling of all instantaneous volatilities "all without a loop" (AWOL), a technique going back at least to Rue (2001) and discussed in more detail in McCausland, Miller, and Pelletier (2011). Doing so reduces correlation of the draws significantly and requires auxiliary finite mixture approximation of the errors as in Kim *et al.* (1998) or Omori, Chib, Shephard, and Nakajima (2007).

In order to avoid the cost of code interpretation within each MCMC iteration, the core computations of the sampler are implemented in C, interfaced to R via the **Rcpp** package (Eddelbuettel and François 2011), where the convenience functions and the user-interface are implemented. This combination allows to make use of the well-established and widely accepted ease-of-use of R and its underlying functional programming paradigm. Moreover, existing frameworks for analyzing MCMC output such as **coda** (Plummer, Best, Cowles, and Vines 2006) as well as high-level visualization tools can easily be used. Last but not least, users with a basic knowledge of R can use the package in a familiar surrounding with a very small entry cost. Nevertheless, despite all these convenience features, the package profits from highly optimized machine code generated by a compiler at package build time, thus providing acceptable runtime even for larger datasets.

A novel and crucial feature of the algorithm implemented in **stochvol** is the usage of an "ancillarity-sufficiency interweaving strategy" (ASIS), which has been brought forward in the general context of state-space models by Yu and Meng (2011). ASIS exploits the fact that for certain parameter constellations, sampling efficiency improves substantially when considering a non-centered version of a state-space model. This fact is commonly known as a reparameterization issue with an entire body of literature attached to it; for an early reference see e.g., Hills and Smith (1992). In the case of the SV model, a move of this kind can be achieved by moving the level μ and/or the volatility σ_η of log-variance from the state equation (2) to the observation equation (1) through a simple reparameterization of the latent process $\boldsymbol{h}$. However, in the case of the SV model, it turns out that no single superior parameterization exists. Rather, for some underlying processes, the standard parameterization yields superior results, while for other processes non-centered versions are better. To overcome this issue, the parameter vector $\boldsymbol{\theta}$ is sampled twice: once in the centered, and once in a non-centered parameterization. This method of "combining best of different worlds" allows for efficient inference regardless of the underlying process with one algorithm. For more details about the algorithm and empirical results concerning sampling efficiency, see Kastner and Frühwirth-Schnatter (2014).

3. The stochvol package

The usual stand-alone approach to fitting SV models with **stochvol** follows the following workflow: (1) Prepare the data, (2) specify the prior distributions and configuration parameters, (3) run the sampler, (4) assess the output and display the results. All these steps will be described in more detail below, along with a worked example. For a stepwise incorporation of SV effects into other MCMC samplers, please see Section 4.

3.1. Preparing the data

The core sampling function `svsample` expects its input data `y` to be a numeric vector of returns without any missing values (`NA`s), and will throw an error if provided with anything else. In case that `y` contains zeros, a warning will be issued and a small offset constant of size `sd(y)/10000` will be added to the squared returns before doing the auxiliary mixture sampling (cf. Omori *et al.* 2007). A common and recommended way of avoiding zero returns is to de-mean the returns beforehand. Below is an example how to prepare data, illustrated with the `exrates` data set[1] which is included in the package. A visualization of one of these time series is displayed in Figure 1.

```
R> set.seed(123)
R> library("stochvol")
R> data("exrates")
R> ret <- logret(exrates$USD, demean=TRUE)
R> par(mfrow=c(2, 1), mar = c(1.9, 1.9, 1.9, .5), mgp = c(2, .6, 0))
R> plot(exrates$date, exrates$USD, type = 'l', main = "Price of 1 EUR in USD")
R> plot(exrates$date[-1], ret, type = 'l', main = "Demeaned log-returns")
```

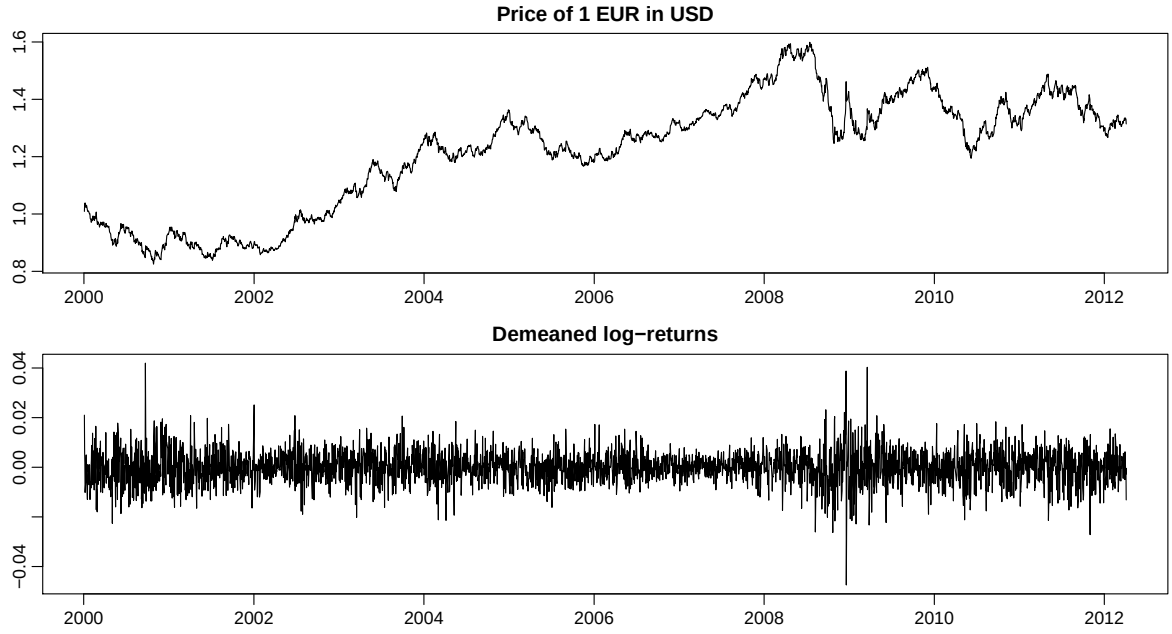

Figure 1: Visualization of EUR-USD exchange rates included in the **stochvol** package.

Additionally to real-world data, **stochvol** has also a built-in data generator `svsim`. This function simply produces realizations of an SV process and returns an object of class `svsim`,

[1]The data set, which as been obtained from the European Central Bank's Statistical Data Warehouse, contains the daily bilateral prices of one Euro in 23 currencies from January 3, 2000, until April 4, 2012. Conversions to New Turkish Lira and Fourth Romanian Leu have been incorporated. See `?exrates` for more information.

which has its own `print`, `summary`, and `plot` methods. Exemplary code using `svsim` is given below, and the particular instance of this simulated series is displayed in Figure 2.

```
R> sim <- svsim(500, mu = -9, phi = 0.99, sigma = 0.1)
R> par(mfrow=c(2, 1))
R> plot(sim)
```

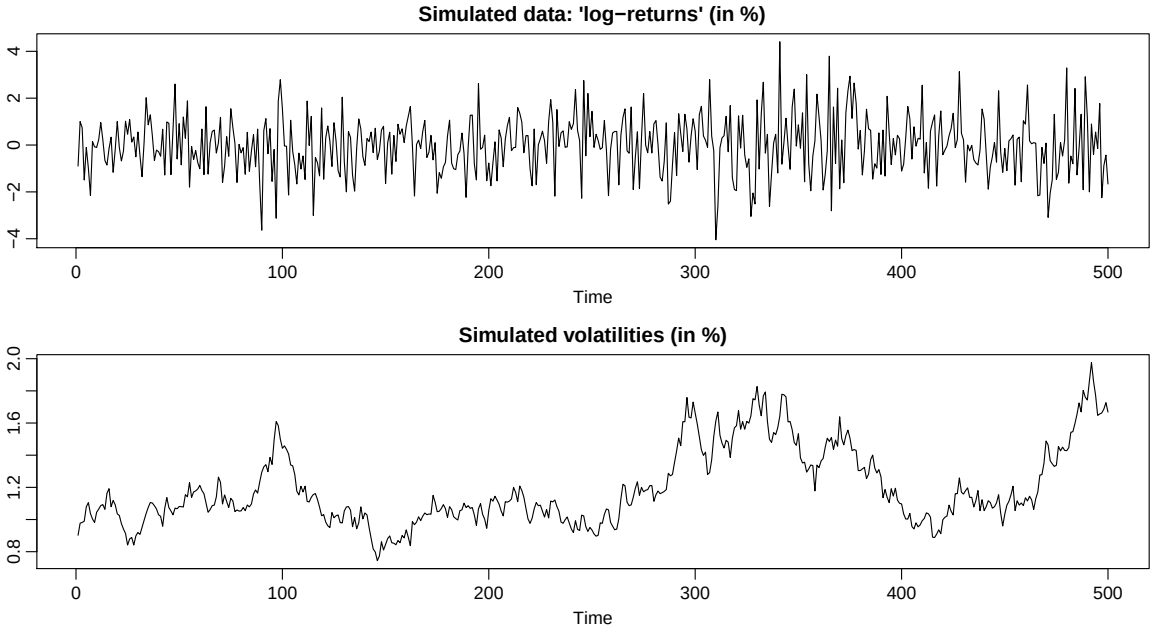

Figure 2: Visualization of a simulated time series as provided by the default `plot` method.

3.2. Specifying prior distributions and configuration parameters

After preparing the data vector `y`, the user needs to specify the prior hyperparameters for the parameter vector $\boldsymbol{\theta} = (\mu, \phi, \sigma_\eta)^\top$ – see also Section 2.2 – and some configuration parameters. The appropriate values are passed to the main sampling function `svsample` as arguments, which are described below.

The argument `priormu` is a vector of length 2, containing mean and standard deviation of the normal prior for the level of the log-variance μ. A common strategy is to choose a vague prior here, e.g., `c(0, 100)`, because the likelihood usually carries enough information about this parameter. If one prefers to use (slightly) informative priors, e.g., to avoid outlier draws of μ, care must be taken whether actual log-returns or *percentage* log-returns are analyzed. Assuming daily data, the former commonly have an unconditional variance of 0.0001 or less and thus the level on the log-scale μ lies around $\log(0.0001) \approx -9$, while the latter have the 100^2-fold unconditional variance (around 1), which implies a level of $\log(1) = 0$. Choices in the literature include `c(0, 10)` (Jacquier, Polson, and Rossi 2004), `c(0, 5)` (Yu 2005), `c(0, sqrt(10))` (Kim *et al.* 1998; Meyer and Yu 2000) or `c(0, 1)` (Omori *et al.* 2007). Note that most of these choices are quite informative and clearly designed for *percentage* log-returns!

For specifying the prior hyperparameters for the persistence of log-variance, ϕ, the argument `priorphi` may be used. It is again a vector of length 2, containing a_0 and b_0 specified in Equation 4. As elaborated in Section 2.2, these values can possibly be quite influential, thus we advise to put some thought into choosing them and study the effect of different choices carefully. The default is currently given through `c(5, 1.5)`, implying a prior mean of 0.54 and a prior standard deviation of 0.31.

The prior variance of log-variance hyperparameter B_{σ_η} may be controlled through `priorsigma`. This argument defaults to `1` if not provided by the user. As discussed in Section 2.2, the exact choice of this value is usually not very influential in typical applications. In general, it should not be picked too small unless there is a very good reason, e.g., explicit prior knowledge, to do so.

For specifying the size of the burn-in, the parameter `burnin` is provided. It is the amount of MCMC iterations that are run but discarded to ensure convergence to the stationary distribution of the chain. The current default value for this parameter is 1000, which has proved to suffice in most situations. Nevertheless, the user is encouraged to check convergence carefully, see Section 3.4 for more details. The amount of iterations which are run after burn-in can be specified through the parameter `draws`, currently defaulting to 10 000. Consequently, the sampler will be run for a total of `burnin + draws` iterations.

Three thinning parameters are available, which all are 1 if not specified otherwise. The first one, `thinpara`, specifies the denominator of the fraction of parameter draws (i.e., draws of $\boldsymbol{\theta}$) that are stored. E.g., if `thinpara` equals 10, every 10th draw is kept. The default parameter thinning value of 1 means that all draws are saved. The second thinning parameter, `thinlatent`, acts in the same way for the latent variables $\boldsymbol{h}$. The third thinning parameter, `thintime`, refers to thinning with respect to the time dimension of the latent volatility. In the case that `thintime` is greater than 1, not all elements of $\boldsymbol{h}$ are stored, e.g., for `thintime` equaling 10, only the draws of $h_1, h_{11}, h_{21}, \ldots$ (and h_0) are kept.

Another configuration argument is `quiet`, which defaults to `FALSE`. If set to `TRUE`, all output during sampling (progress bar, status messages) is omitted. The arguments `startpara` and `startlatent` are optional starting values for the parameter vector $\boldsymbol{\theta}$ and the latent variables $\boldsymbol{h}$, respectively. All other configuration parameters are summarized in the argument `expert`, because it is not very likely that the end-user needs to mess with the defaults.[2] Please refer to the package documentation in combination with Kastner and Frühwirth-Schnatter (2014) for details.

Any further arguments (`...`) will be forwarded to `updatesummary`, controlling the type of summary statistics that are calculated for the posterior draws.

3.3. Running the sampler

At the heart of the package **stochvol** lies the function `svsample`, which serves as an R-wrapper for the actual sampler coded in C. Exemplary usage of this function is given in the code snipped below, along with the default output.

[2]Examples of configurations that can be changed with the `expert`-argument include the specification of the (baseline) parameterization (*centered* or *noncentered*) and the possibility to turn off interweaving. Moreover, some algorithmic details such as the number of blocks used for the parameter updates or the possibility of using a random walk Metropolis-Hastings proposal (instead of the default independence proposal) can be found here.

```
R> res <- svsample(ret, priormu = c(-10, 1), priorphi = c(20, 1.1),
+                  priorsigma = .1)

Calling GIS_C MCMC sampler with 11000 iter. Series length T = 3139.

  0% [++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++] 100%

Timing (elapsed): 17.935 seconds.
613 iterations per second.

Converting results to coda objects... Done!
Summarizing posterior draws... Done!
```

As can be seen, this function calls the main MCMC sampler and converts its output to **coda**-compatible objects. Moreover, some summary statistics for the posterior draws are calculated. The return value of `svsample` is an object of type `svdraws`, which is a named list with eight elements, holding (1) the parameter draws in `para`, (2) the latent log-volatilities in `latent`, (3) the initial latent log-volatility draw in `latent0`, (4) the data provided in `y`, (5) the sampling runtime in `runtime`, (6) the prior hyperparameters in `priors`, (7) the thinning values in `thinning`, and (8) summary statistics of these draws, alongside some common transformations thereof, in `summary`.

3.4. Assessing the output and displaying the results

Following common practice, `print` and `summary` methods are available for `svdraws` objects. Each of these has two optional parameters, `showpara` and `showlatent`, specifying which output should be displayed. If `showpara` is `TRUE` (the default), values/summaries of the parameter draws are shown. If `showlatent` is `TRUE` (the default), values/summaries of the latent variable draws are shown. In the example below, the summary for the parameter draws only is displayed.

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

Summary of 10000 MCMC draws after a burn-in of 1000.

Prior distributions:
mu        ~ Normal(mean = -10, sd = 1)
(phi+1)/2 ~ Beta(a0 = 20, b0 = 1.1)
sigma^2   ~ 0.1 * Chisq(df = 1)

Posterior draws of parameters (thinning = 1):
              mean      sd       5%      50%     95%  ESS
mu         -10.1308 0.22806 -10.4661 -10.1345 -9.7746 4537
phi          0.9936 0.00282   0.9886   0.9938  0.9977  386
sigma        0.0654 0.01005   0.0510   0.0646  0.0827  142
exp(mu/2)    0.0064 0.00076   0.0053   0.0063  0.0075 4537
sigma^2      0.0044 0.00139   0.0026   0.0042  0.0068  142
```

There are several plotting functions specifically designed for objects of class `svsample`, which will be described in the following paragraphs.

(1) `volplot`: Plots posterior quantiles of the latent volatilities in percent, i.e., $100 \exp(h_t/2)$, over time. Apart from the mandatory `svsample`-object itself, this function takes several optional arguments. Only some will be mentioned here; for an exhaustive list please see the corresponding help document accessible through `?volplot` or `help(volplot)`. Selected optional arguments that are commonly used include `forecast` for n-step-ahead volatility prediction, `dates` for date-labels on the x-axis, alongside some graphical parameters. The code snipped below shows a typical example and Figure 3 displays its output.

```
R> volplot(res, forecast = 100, dates = exrates$date[-1])
```

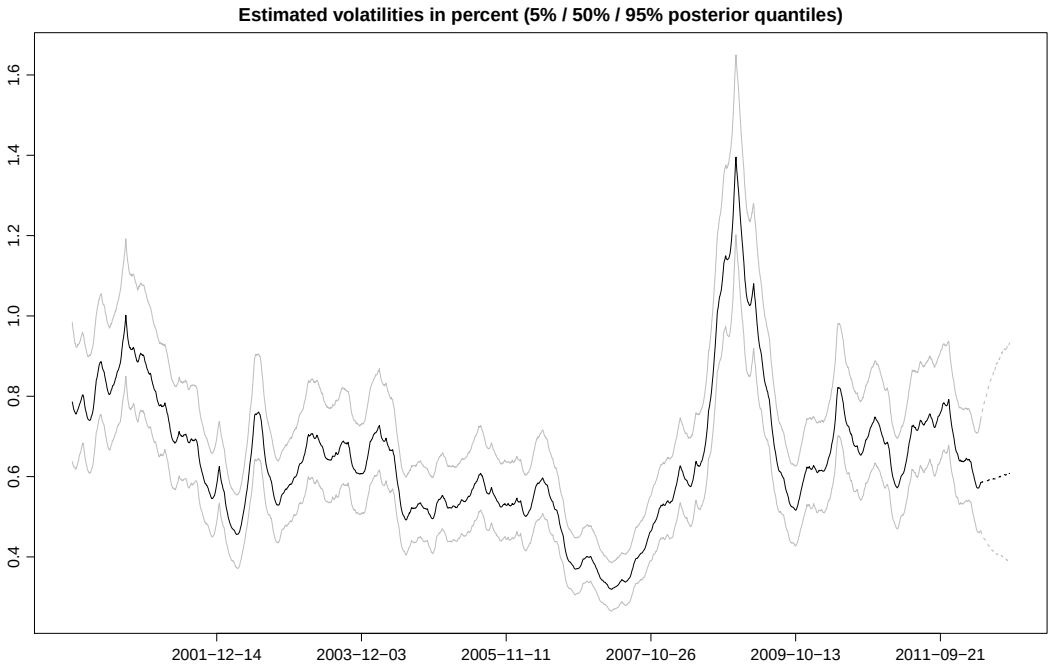

Figure 3: Visualization of estimated contemporaneous volatilities of EUR-USD exchange rates, as provided by `volplot`. If not specified otherwise, posterior medians and 5%/95% quantiles are plotted. The dotted lines at the right side indicate predicted future volatilities.

In case the user wants to display different posterior quantiles, the `updatesummary` function has to be called first. See the code below for an example and Figure 4 for the corresponding plot.

```
R> res <- updatesummary(res, quantiles = c(.01, .1, .5, .9, .99))
R> volplot(res, forecast = 100, dates = exrates$date[-1])
```

(2) `paratraceplot`: Displays trace plots for the parameters contained in $\boldsymbol{\theta}$. Note that the burn-in has already been discarded. Figure 5 shows an example.

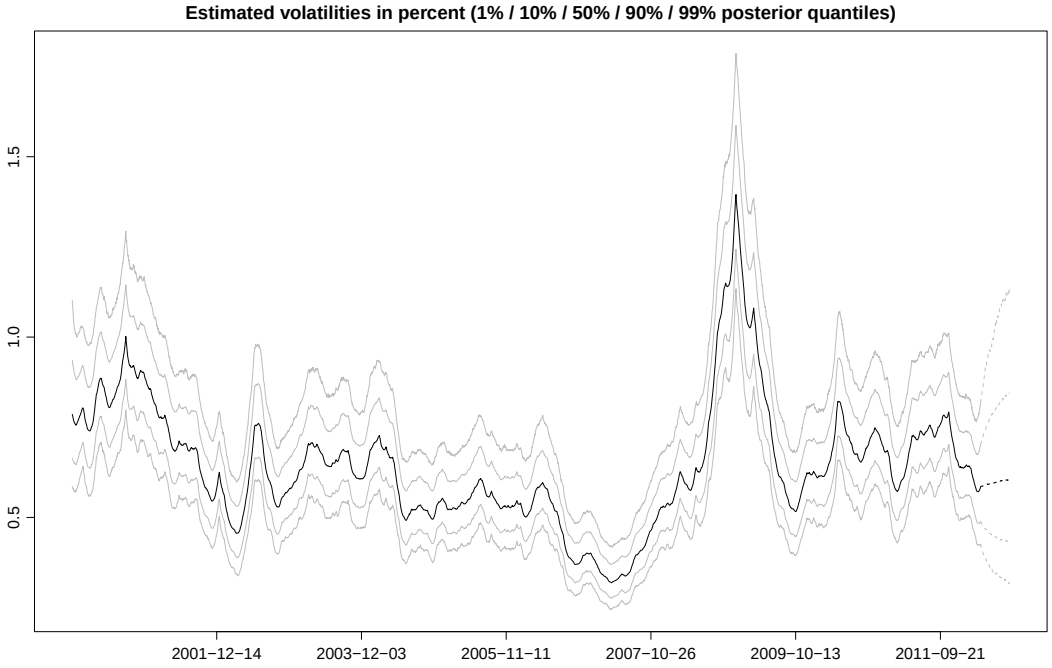

Figure 4: As above, now with medians (black line) and 1%/10%/90%/99% quantiles (gray lines). This behavior can be achieved through a preceding call of `updatesummary`.

```
R> par(mfrow = c(3, 1))
R> paratraceplot(res)
```

(3) `paradensplot`: Displays a kernel density estimate for the parameters contained in $\boldsymbol{\theta}$. If the argument `showobs` is `TRUE` (which is the default), individual posterior draws are indicated through a *rug*, i.e., short vertical lines at the x-axis. For quicker drawing of large posterior samples, this argument should be set to `FALSE`. If the argument `showprior` is `TRUE` (which is the default), the prior distribution is indicated through a dashed gray line. Figure 6 shows an example output for the EUR-USD exchange rates obtained from the `exrates` dataset.

```
R> par(mfrow = c(1, 3))
R> paradensplot(res)
```

The generic `plot` method for `svdraws` objects combines all above plots into one plot. All arguments described above can be used. See `?plot.svsample` for an exhaustive summary of possible arguments and Figure 7 for an example.

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

For extracting standardized residuals, the `residuals`/`resid` method can be used on a given `svdraws` object. With the optional argument `type`, the type of summary statistic may be

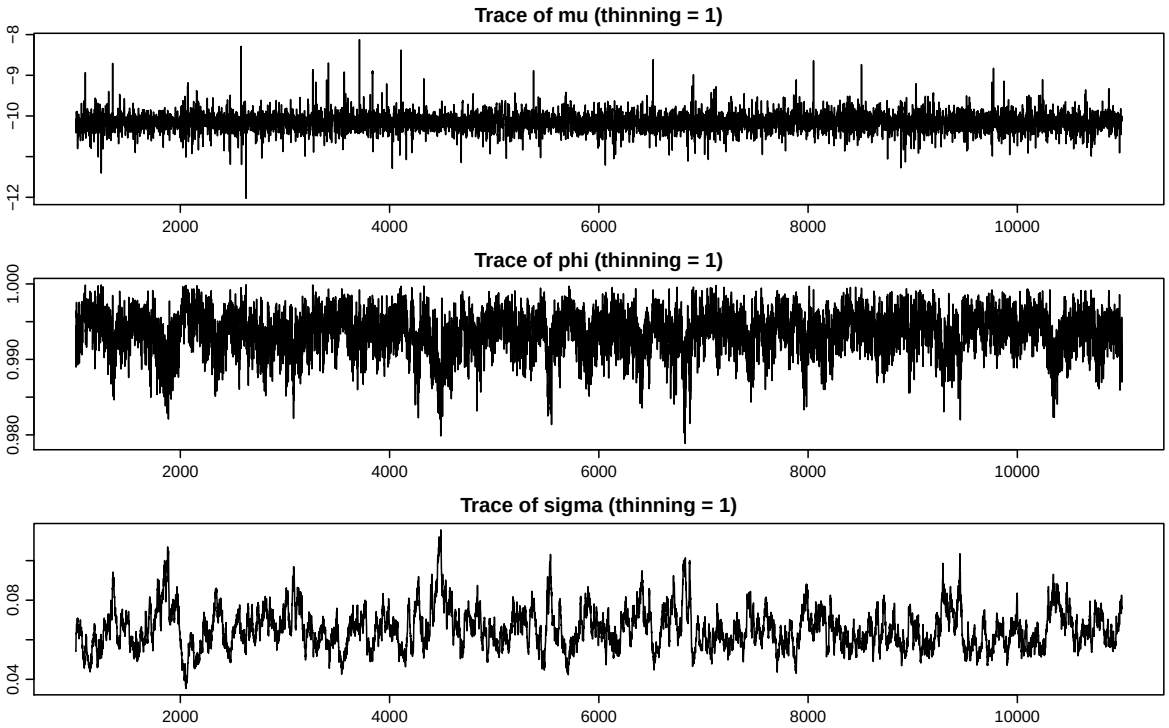

Figure 5: Trace plots of posterior draws for the parameters μ, ϕ, σ_η.

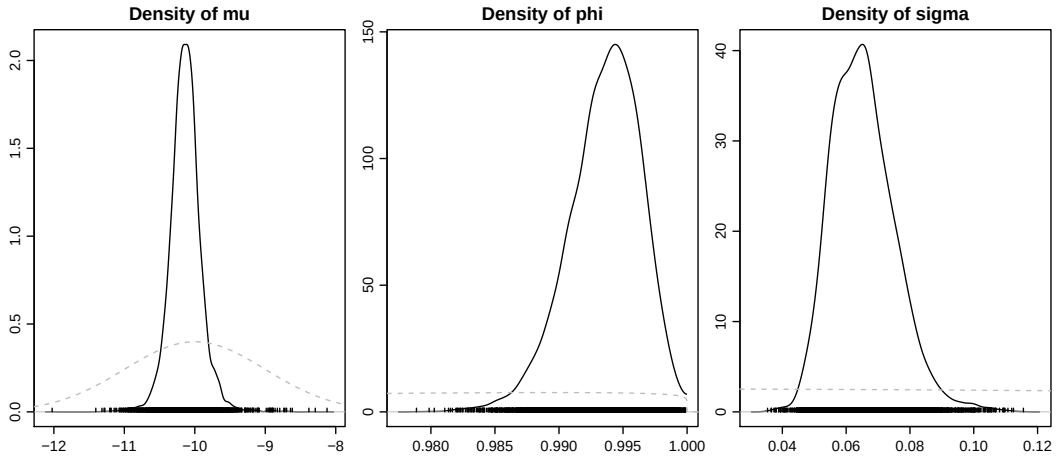

Figure 6: Posterior density estimates (black solid lines) along with prior densities (dashed gray lines). Individual posterior draws are indicated by the underlying rug.

specified. Currently, `type` is allowed to be either `"mean"` or `"median"`, where the former corresponds to the default value. This method returns a real vector of class `svresid`, which contains the requested summary statistic of standardized residuals for each point in time. There is also a `plot` method available, providing the option of comparing the standardized

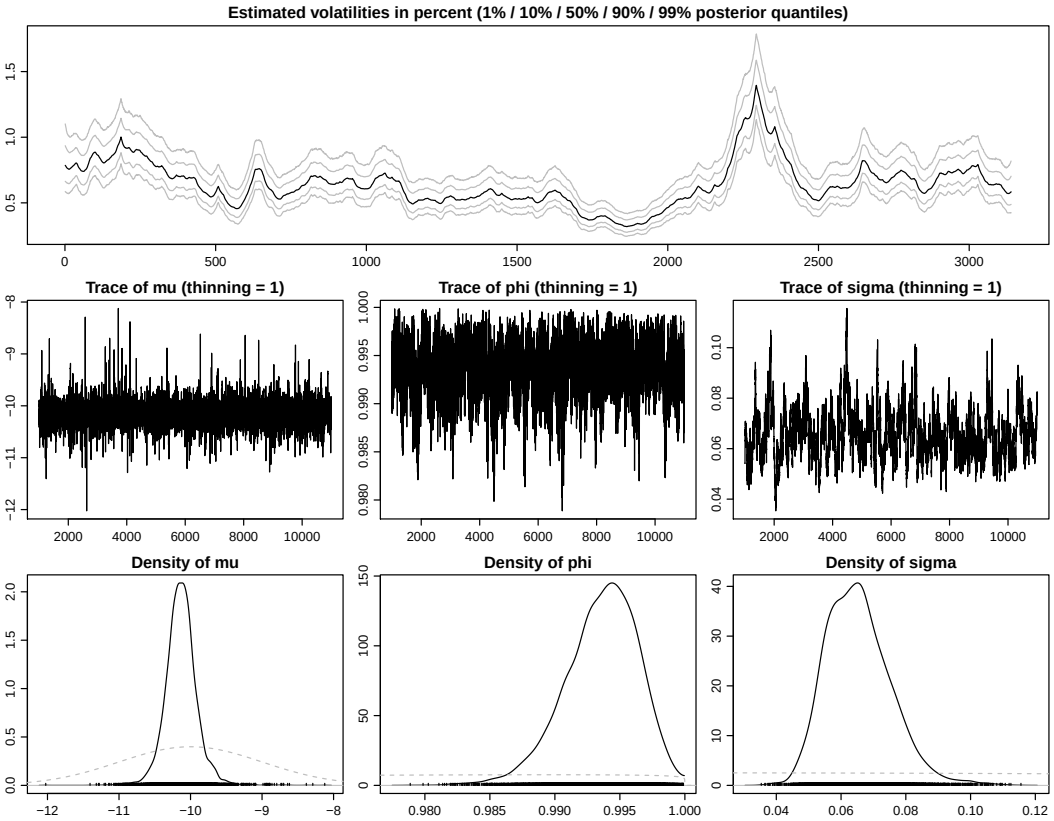

Figure 7: Illustration of the default `plot` method for `svdraws`-objects. This visualization combines `volplot` (Figure 4), `traceplot` (Figure 5), and `paradensplot` (Figure 6) into one single plot.

residuals to the original data when given through the argument `origdata`. See the code below for an example and Figure 8 for the corresponding output.

```
R> myresid <- resid(res)
R> plot(myresid, ret)
```

4. Using stochvol within other samplers

We demonstrate how the **stochvol** package can be used to incorporate stochastic volatility into any given MCMC sampler. For the sake of simplicity, we explain this procedure with the help of the Bayesian normal linear model with T observations and $k = p - 1$ predictors, given through

$$\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\Sigma} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\Sigma})\,. \tag{5}$$

Here, $\boldsymbol{y}$ denotes the $T \times 1$ vector of responses, $\boldsymbol{X}$ is the $T \times p$ design matrix containing ones in the first column and the predictors in the others, and $\boldsymbol{\beta}$ stands for the $p \times 1$ vector of regression

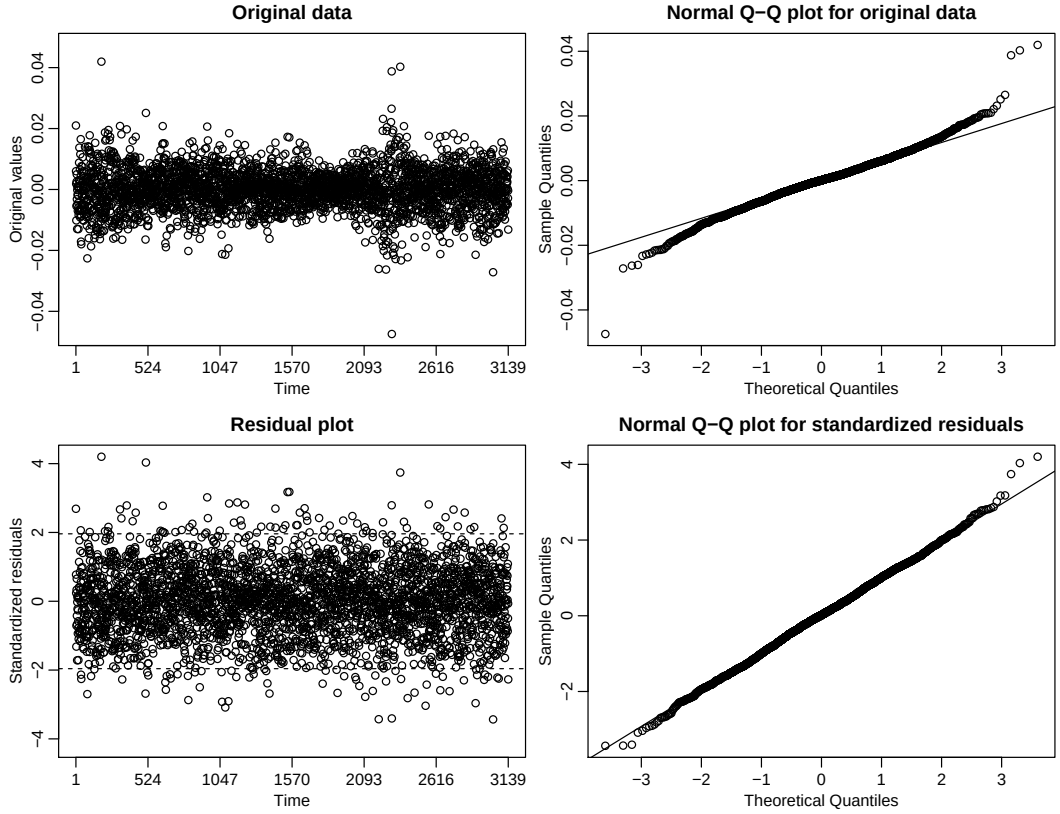

Figure 8: Mean standardized residual plots for assessing the model fit, as provided by the corresponding `plot` method. The dashed lines in the bottom left panel indicate the 2.5%/97.5% quantiles of the standard normal distribution.

coefficients. In the following sections, we will discuss two specifications of the $T \times T$ error covariance matrix $\boldsymbol{\Sigma}$.

4.1. The Bayesian normal linear model with homoskedastic errors

The arguably simplest specification of the error covariance matrix in Equation 5 is given by $\boldsymbol{\Sigma} \equiv \sigma_\epsilon^2 \boldsymbol{I}$, where $\boldsymbol{I}$ denotes the T-dimensional unit matrix. This specification is used in many applications and commonly referred to as the linear regression model with homoskedastic errors. To keep things simple, let model parameters $\boldsymbol{\beta}$ and σ_ϵ^2 be equipped with the usual conjugate prior $p(\boldsymbol{\beta}, \sigma_\epsilon^2) = p(\boldsymbol{\beta}|\sigma_\epsilon^2)p(\sigma_\epsilon^2)$, where

$$\begin{aligned} \boldsymbol{\beta}|\sigma_\epsilon^2 &\sim \mathcal{N}\left(\boldsymbol{b}_0, \sigma_\epsilon^2 \boldsymbol{B}_0\right), \\ \sigma_\epsilon^2 &\sim \mathcal{G}^{-1}(c_0, C_0). \end{aligned}$$

A commonly used Gibbs-sampler for drawing from the posterior distribution of this model is given by sampling in turn from the full conditional distributions $\boldsymbol{\beta}|\boldsymbol{y}, \sigma_\epsilon^2 \sim \mathcal{N}(\boldsymbol{b}_{\boldsymbol{T}}, \boldsymbol{B}_{\boldsymbol{T}})$ with

$$\boldsymbol{b}_{\boldsymbol{T}} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{B}_0^{-1}\right)^{-1} \left(\boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{B}_0^{-1} \boldsymbol{b}_0\right), \qquad \boldsymbol{B}_{\boldsymbol{T}} = \sigma_\epsilon^2 \left(\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{B}_0^{-1}\right)^{-1},$$

and $\sigma_\epsilon^2|\boldsymbol{y}, \boldsymbol{\beta} \sim \mathcal{G}^{-1}(c_T, C_T)$ with

$$c_T = c_0 + \frac{T}{2} + \frac{p}{2}, \qquad C_T = C_0 + \frac{1}{2}\left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{b}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{b}_0)\right).$$

In R, this can straightforwardly be coded as follows:

- Set seed to make results reproducible and simulate some data:

```
R> set.seed(123456)
R> T <- 1000
R> beta.true <- c(.1, .5)
R> sigma.true <- 0.01
R> X <- matrix(c(rep(1, T), rnorm(T, sd = sigma.true)), nrow = T)
R> y <- rnorm(T, X %*% beta.true, sigma.true)
```

- Specify configuration parameters and prior values:

```
R> draws <- 5000
R> burnin <- 100
R> b0 <- matrix(c(0, 0), nrow = ncol(X))
R> B0inv <- diag(c(10^-10, 10^-10))
R> c0 <- 0.001
R> C0 <- 0.001
```

- Pre-calculate some values outside the main MCMC loop:

```
R> p <- ncol(X)
R> preCov <- solve(crossprod(X) + B0inv)
R> preMean <- preCov %*% (crossprod(X, y) + B0inv %*% b0)
R> preDf <- c0 + T/2 + p/2
```

- Assign some storage space for holding the draws and set an initial value for σ_ϵ^2:

```
R> draws1 <- matrix(NA_real_, nrow = draws, ncol = p + 1)
R> colnames(draws1) <- c(paste("beta", 0:(p-1), sep='_'), "sigma")
R> sigma2draw <- 1
```

- Run the main sampler: Iteratively draw from the conditional bivariate Gaussian distribution $\boldsymbol{\beta}|\boldsymbol{y}, \sigma_\epsilon^2$, e.g., through the use of **mvtnorm** (Genz *et al.* 2013), and the conditional Inverse Gamma distribution $\sigma_\epsilon^2|\boldsymbol{y}, \boldsymbol{\beta}$.

```
R> for (i in -(burnin-1):draws) {
+   betadraw <- as.numeric(mvtnorm::rmvnorm(1, preMean, sigma2draw*preCov))
+   tmp <- C0 + .5*(crossprod(y - X%*%betadraw) +
+                   crossprod((betadraw - b0), B0inv) %*% (betadraw - b0))
+   sigma2draw <- 1/rgamma(1, preDf, rate = tmp)
+   if (i > 0) draws1[i,] <- c(betadraw, sqrt(sigma2draw))
+ }
```

- Finally, visualize the posterior draws:

```
R> par(mar = c(3.1, 1.8, 1.9, .5), mgp = c(1.8, .6, 0))
R> plot(coda::mcmc(draws1))
```

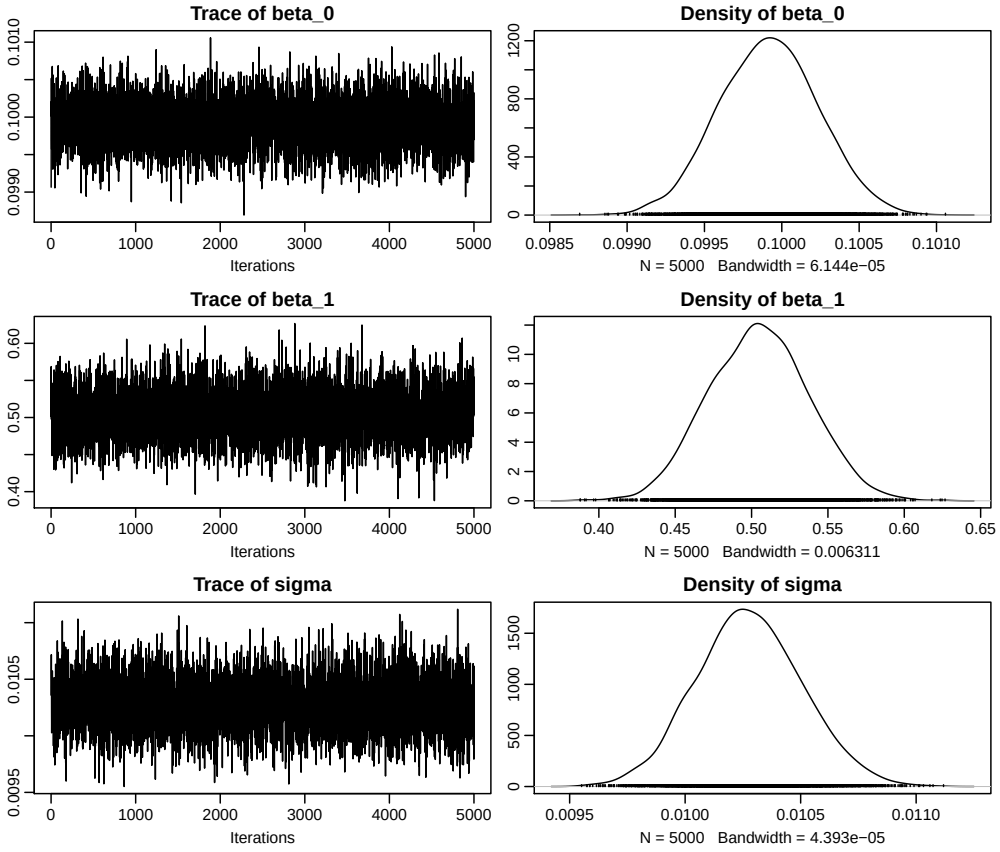

Figure 9: Trace plots and kernel density estimates for some draws from the marginal posterior distributions in the regression model with heteroskedastic errors. Underlying data is simulated with $\boldsymbol{\beta}^{\text{true}} = (0.1, 0.5)^\top$, $\sigma_\epsilon^{\text{true}} = 0.01$, $T = 1000$.

```
R> colMeans(draws1)

    beta_0     beta_1      sigma
0.09991649 0.50472433 0.01027775
```

4.2. The Bayesian normal linear model with SV errors

Instead of homoskedastic errors, we now specify the error covariance matrix in Equation 5 to be $\boldsymbol{\Sigma} \equiv \text{diag}(e^{h_1}, \ldots, e^{h_T})$, thus introducing nonlinear dependence between the observations due to the AR(1)-nature of $\boldsymbol{h}$. Instead of cooking up an entire new sampler, we adapt the code from above utilizing the **stochvol** package. To do so, we simply replace the sampling step of σ_ϵ^2 from an Inverse-Gamma distribution by a sampling step of $\boldsymbol{\theta}$ and $\boldsymbol{h}$ through a call to `.svsample`. This "dotted" function is a minimal-overhead version of the regular `svsample`. It

provides the full sampling functionality of the "non-dotted" version, but has slightly different default values, a simplified return value structure and does not perform costly input checks. Thus, it is optimized for repeated calls but needs to be used with proper care. Note that the current draws of the variables need to be passed to the function through `startpara` and `startlatent`.

- Simulate some data:

```
R> mu.true <- log(sigma.true^2)
R> phi.true <- 0.97
R> volvol.true <- 0.3
R> simresid <- svsim(T, mu = mu.true, phi = phi.true, sigma = volvol.true)
R> y <- X %*% beta.true + simresid$y
```

- Specify configuration parameters and prior values:

```
R> draws <- 50000
R> burnin <- 1000
R> thinning <- 10
R> priormu <- c(-10, 2)
R> priorphi <- c(20, 1.5)
R> priorsigma <- 1
```

- Assign some storage space for holding the draws and set initial values:

```
R> draws2 <- matrix(NA_real_, nrow = floor(draws/thinning), ncol = 3 + T + p)
R> colnames(draws2) <- c("mu", "phi", "sigma", paste("beta", 0:(p-1), sep='_'),
+                        paste("h", 1:T, sep='_'))
R> betadraw <- c(0, 0)
R> svdraw <- list(para = c(mu = -10, phi = .9, sigma = .2), latent = rep(-10, T))
```

- Run the main sampler: Iteratively draw the latent volatilities (and AR-parameters) through conditioning on the regression parameters and calling `.svsample`, and the regression parameters through conditioning on the latent volatilities.

```
R> for (i in -(burnin-1):draws) {
+   ytilde <- y - X %*% betadraw
+   svdraw <- .svsample(ytilde, startpara=para(svdraw),
+                       startlatent=latent(svdraw), priormu=priormu,
+                       priorphi=priorphi, priorsigma=priorsigma)
+   normalizer <- as.numeric(exp(-latent(svdraw)/2))
+   Xnew <- X * normalizer
+   ynew <- y * normalizer
+   Sigma <- solve(crossprod(Xnew) + B0inv)
+   mu <- Sigma %*% (crossprod(Xnew, ynew) + B0inv %*% b0)
+   betadraw <- as.numeric(mvtnorm::rmvnorm(1, mu, Sigma))
+   if (i > 0 & i %% thinning == 0) {
+    draws2[i/thinning, 1:3] <- para(svdraw)
+    draws2[i/thinning, 4:5] <- betadraw
```

```
+    draws2[i/thinning, 6:(T+5)] <- latent(svdraw)
+   }
+  }
```

- Finally, visualize (some) posterior draws:

```
R> plot(coda::mcmc(draws2[,4:7]))
```

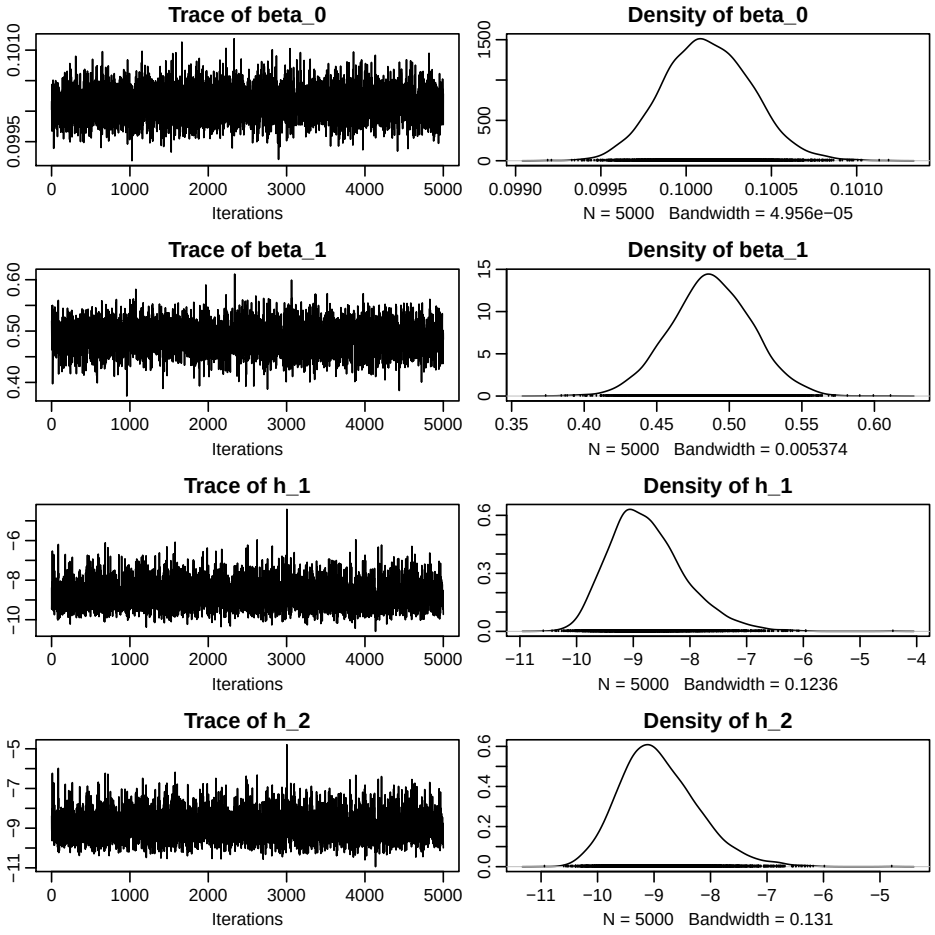

Figure 10: Trace plots and kernel density estimates in the regression model with heteroskedastic errors. Data is simulated with $\boldsymbol{\beta}^{\text{true}} = (0.1, 0.5)^\top$, $h_1^{\text{true}} = -8.28$, $h_2^{\text{true}} = -8.50$, $T = 1000$.

```
R> colMeans(draws2[,4:8])

    beta_0     beta_1        h_1        h_2        h_3
 0.1001254  0.4873015 -8.7512804 -8.9052584 -9.0276110
```

5. Real-world example

In the following, we aim for a comparison of the performance of the Bayesian normal linear

model with homoskedastic errors from Section 4.1 with the Bayesian normal linear model with SV errors from Section 4.2 using the `exrates` data set introduced in Section 3.1.

5.1. Model setup

We again use the daily price of 1 EUR in USD from January 3, 2000, until April 4, 2012, denoted by $\boldsymbol{p} = (p_1, p_2, \ldots, p_T)^\top$. This time however, instead of using log-returns, we investigate the development of raw prices by regression: Let $\boldsymbol{y}$ contain all raw observations except the very first one, and let $\boldsymbol{X}$ denote the design matrix containing ones in the first column and lagged raw prices in the second, i.e.

$$\boldsymbol{y} = \begin{pmatrix} p_2 \\ p_3 \\ \vdots \\ p_T \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & p_1 \\ 1 & p_2 \\ \vdots & \vdots \\ 1 & p_{T-1} \end{pmatrix}.$$

Clearly, we expect the posterior distribution of $\boldsymbol{\beta}$ to spread around $(0, 1)^\top$, which corresponds to a random walk. A scatterplot of p_t against p_{t+1}, displayed in Figure 11, confirms this.

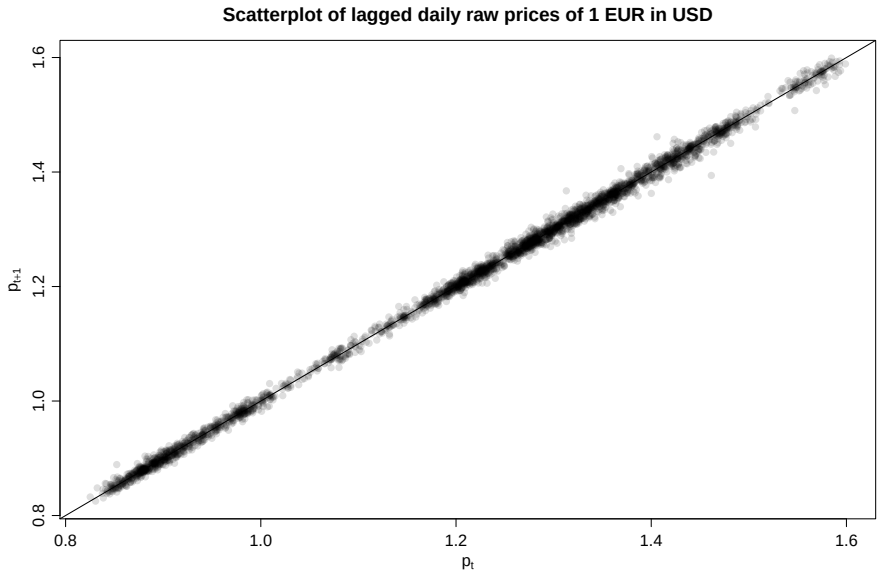

Figure 11: Scatterplot of daily raw prices at time t against daily raw prices at time $t+1$. The solid line indicates the identity function $f(x) = x$.

5.2. Posterior inference

We run both samplers for 100 000 iterations and discard the first 10 000 draws as burn-in. Prior hyperparameters are chosen as follows: $\boldsymbol{b}_0 = (0, 0)^\top$, $\boldsymbol{B}_0 = \mathrm{diag}(10^{10}, 10^{10})$, $c_0 = C_0 = 0.001$, $b_\mu = -10$, $B_\mu = 2$, $a_0 = 20$, $b_0 = 1.5$, $B_{\sigma_\eta} = 1$. However, due to the length of the dataset (and its obvious heteroskedasticity), the exact prior specification is not very influential. The two samplers yield slightly different posteriors for $\boldsymbol{\beta}$, visualized in Figure 12, where both the

marginal posterior densities $p(\beta_0|\boldsymbol{y})$ and $p(\beta_1|\boldsymbol{y})$ as well as a scatterplot of draws from the joint posterior $\boldsymbol{\beta}|\boldsymbol{y}$ are displayed.

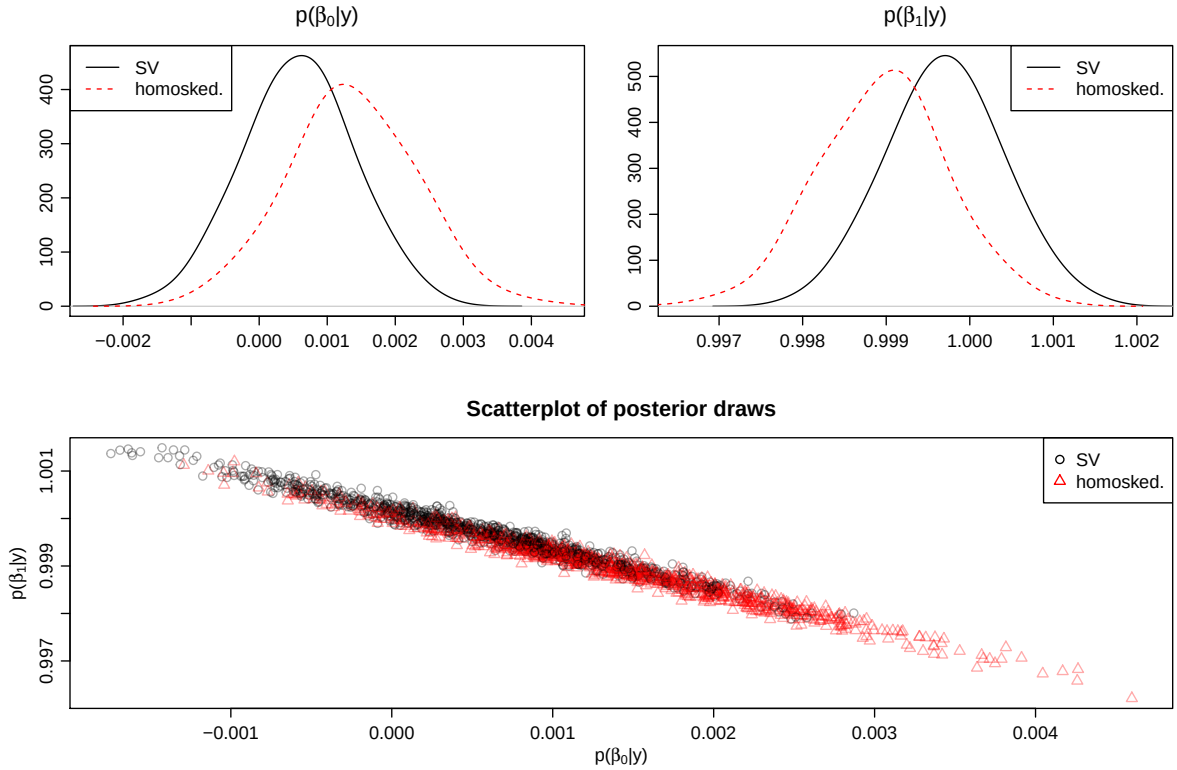

Figure 12: Visualization of the posterior distributions $\boldsymbol{\beta}|\boldsymbol{y}$ for the model with SV residuals and the model with homoskedastic errors. Top panels: kernel density estimate of the univariate posterior marginal distributions. Bottom panel: bivariate scatterplot of posterior draws.

To assess the model fit, mean standardized resdiduals for both samplers are depicted in Figure 13. It stands out that the model with homoskedastic errors shows deficiencies in terms of heavy correlation amongst the residuals. This can clearly be seen in the top left panel, where mean standardized residuals are plotted against time. The bottom left panel shows the same plot for the model with SV errors, where this effect practically vanishes. Moreover, in the model with homoskedastic errors, the normality assumption about the unconditional error distribution is clearly violated. This can be seen by inspecting the quantile-quantile plot in the top right panel, where observed residuals show much heavier tails than one would expect from a normal distribution. On the contrary, standardized residuals obtained from the model with SV errors align almost perfectly.

5.3. Predictive performance and model fit

Within a Bayesian framework, a natural way of assessing the predictive performance of a given model is through its *predictive density* (sometimes also referred to as *posterior predictive distribution*). Collecting all *unobservables*, i.e., parameters and possible latent variables, in a

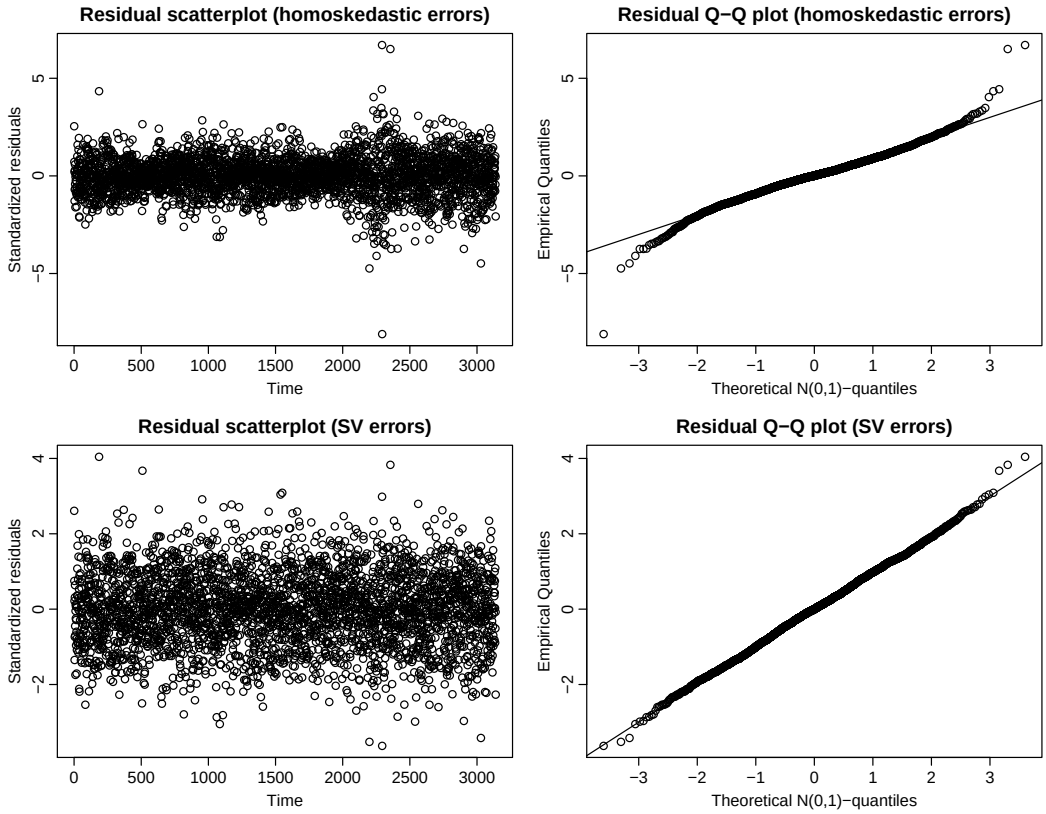

Figure 13: Visualization of mean standardized residuals. Top left panel shows a scatterplot against time for the model with homoskedastic errors, bottom left panel shows this plot for the model with SV errors. Quantile-Quantile plots of empirical quantiles against expected quantiles from a $\mathcal{N}(0,1)$-distribution are displayed on the panels on the right-hand side.

single vector $\boldsymbol{\kappa}$, it is given through

$$p(y_{t+1}|\boldsymbol{y}^o_{[1:t]}) = \int_{\boldsymbol{K}} p(y_{t+1}|\boldsymbol{y}^o_{[1:t]}, \boldsymbol{\kappa}) \times p(\boldsymbol{\kappa}|\boldsymbol{y}^o_{[1:t]})\, \mathrm{d}\boldsymbol{\kappa}. \tag{6}$$

Note that for the model with homoskedastic errors, $\boldsymbol{\kappa} = (\boldsymbol{\beta}, \sigma_\epsilon)^\top$, while for the model with SV errors, $\boldsymbol{\kappa} = (\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{h})^\top$. By using the superscript o in $\boldsymbol{y}^o_{[1:t]}$, we follow Geweke and Amisano (2010) and denote *ex post* realizations (observations) for the set of points in time $\{1, 2, \ldots, t\}$ of the *ex ante* random values $\boldsymbol{y}_{[1:t]} = (y_1, y_2, \ldots, y_t)^\top$. Integration is carried out over $\boldsymbol{K}$, which simply stands for the space of all possible values for $\boldsymbol{\kappa}$. Equation 6 can be viewed as the integral of the likelihood function over the joint posterior distribution of the unobservables $\boldsymbol{\kappa}$. Thus, it can be interpreted as the predictive density for a future value y_{t+1} after integrating out the uncertainty about $\boldsymbol{\kappa}$, conditional on the history $\boldsymbol{y}^o_{[1:t]}$.

In the SV errors case, Equation 6 is a $(T+p+3)$-dimensional integral which cannot be solved analytically. Nevertheless, it may be evaluated at an arbitrary point x through Monte Carlo

integration:

$$p(x|\boldsymbol{y}^o_{[1:t]}) \approx \frac{1}{M}\sum_{m=1}^{M} p(x|\boldsymbol{y}^o_{[1:t]}, \boldsymbol{\kappa}^{(m)}_{[1:t]}), \tag{7}$$

where $\boldsymbol{\kappa}^{(m)}_{[1:t]}$ stands for the m^{th} draw from the respective posterior distribution up to time t. If Equation 7 is evaluated at $x = y^o_{t+1}$, we refer to it as the (one-step-ahead) *predicitve likelihood* (at time $t+1$) denoted PL_{t+1}. Moreover, draws from (6) can be obtained by simulating values $y^{(m)}_{t+1}$ from the distribution given through the density $p(y_{t+1}|\boldsymbol{y}^o_{[1:t]}, \boldsymbol{\kappa}^{(m)}_{[1:t]})$, the summands of Equation 7.

For model at hand, the predictive density and likelihood can thus be computed through the following

Algorithm 1 (Predictive density and likelihood evaluation at time $t+1$)

1. *Reduce the data set to a training set* $\boldsymbol{y}^o_{[1:t]} = (y^o_1, \ldots, y^o_t)^\top$.
2. *Run the posterior sampler using data from the training set only to obtain M posterior draws* $\boldsymbol{\kappa}^{(m)}_{[1:t]}$.

(3.) Needed for the SV model only: Simulate M values from the conditional distribution $h_{t+1,[1:t]}|\boldsymbol{y}^o_{[1:t]}, \boldsymbol{\kappa}_{[1:t]}$ *by drawing* $h^{(m)}_{t+1,[1:t]}$ *from a normal distribution with mean* $\mu^{(m)}_{[1:t]} + \phi^{(m)}_{[1:t]}(h^{(m)}_{t,[1:t]} - \mu^{(m)}_{[1:t]})$ *and standard deviation* $\sigma^{(m)}_{\eta,[1:t]}$ *for* $m = 1, \ldots, M$.

4a. To obtain PL_{t+1}, average over M densities of normal distributions with mean $(1, y^o_t) \times \boldsymbol{\beta}^{(m)}_{[1:t]}$ *and standard deviation* $\exp\{h^{(m)}_{t+1,[1:t]}/2\}$ *(SV model) or* $\sigma^{(m)}_{\epsilon,[1:t]}$ *(homoskedastic model), each evaluated at* y^o_{t+1}.

4b. To obtain M draws from the predictive distribution, simulate from a normal distribution with mean $(1, y^o_t) \times \boldsymbol{\beta}^{(m)}_{[1:t]}$ *and standard deviation* $\exp\{h^{(m)}_{t+1,[1:t]}/2\}$ *(SV model) or* $\sigma^{(m)}_{\epsilon,[1:t]}$ *(homoskedastic model) for* $m = 1, \ldots, M$.

To avoid strong dependence on the prior, the first 1000 days are used as training set only and the evaluation of the predictive distribution starts at $t = 1001$, corresponding to December 4, 2003. The results for both models are displayed in Figure 14. In the top panel, the observed series along with the 98% one-day-ahead predictive intervals are displayed. The bottom panel shows the log one-day-ahead predictive likelihood.

Figure 15 displays a zoomed-in version, depicting only the time span from January 2008 until August 2009. Note that while at the beginning of 2008 both credible intervals are very similar, there is a substantial difference one year later, where SV intervals become around twice as large compared to the corresponding homoskedastic analogs. According to the values of the log predictive likelihoods, SV errors can handle the inflated volatility during that time substantially better. Throughout 2009, the width of the intervals as well as the predictive likelihoods consolidate again.

It is worth pointing out that log predictive likelihoods also carry an intrinsic connection to the log *marginal likelihood*, defined through

$$\log ML = \log p(\boldsymbol{y}^o) = \log \int_{\boldsymbol{K}} p(\boldsymbol{y}^o|\boldsymbol{\kappa}) \times p(\boldsymbol{\kappa})\, \mathrm{d}\boldsymbol{\kappa}. \tag{8}$$

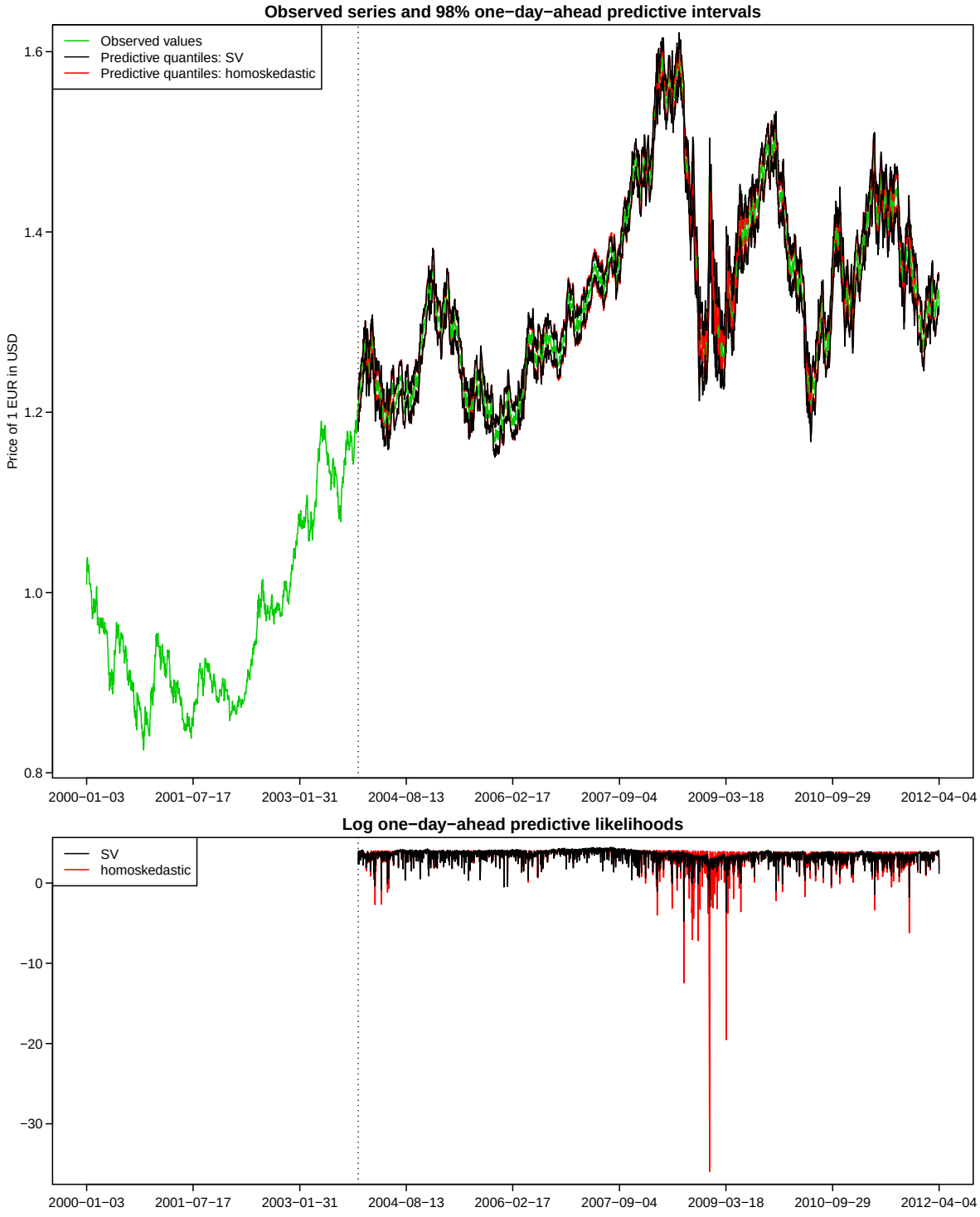

Figure 14: Top panel: Observed series (green) and symmetrical 98% one-day-ahead predictive intervals for the model with homoskedastic errors (red) and the model with SV errors (black). Bottom panel: Log one-day-ahead predictive likelihoods for both models.

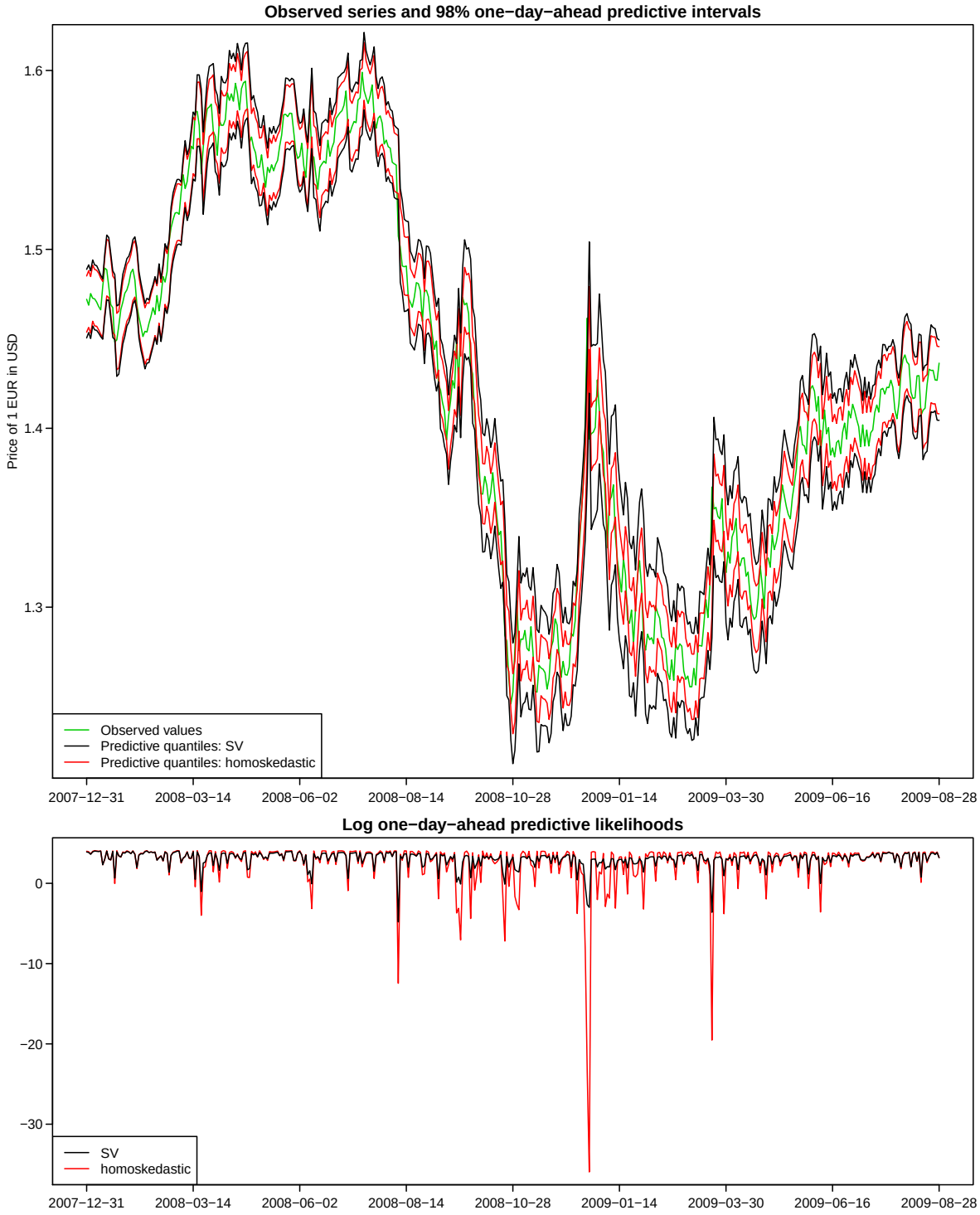

Figure 15: Zoomed version of Figure 14, showing only the period from January 2008 until August 2009. This time span is chosen to includes the beginning of the financial crisis. During that phase, predictive performance of the model with homoskedastic errors deteriorates substantially, while SV errors can capture the inflated volatility much better.

This real number corresponds to the logarithm of the normalizing constant in the denominator of Bayes' law and is often used for evaluating model evidence. It can straightforwardly be decomposed into the sum of the one-step-ahead logarithms of the predictive likelihoods:

$$\log ML = \log p(\boldsymbol{y}^o) = \log \prod_{t=1}^{T} p(y_t^o | \boldsymbol{y}_{[1:t-1]}^o) = \sum_{t=1}^{T} \log PL_t.$$

Thus, Algorithm 1 provides a conceptually simple way of computing the marginal likelihood. However, these computations are quite costly in terms of CPU time, as they require an individual model fit for each of the T points in time. On the other hand, due to the embarrassingly parallel nature of the task and because of today's comparably easy access to parallel computing environments, this burden becomes easily manageable. E.g., the computations for the analysis in this paper have been conducted in less than one hour, using 25 IBM dx360M3 nodes within a cluster of workstations. Implementation in R was achieved through the packages **parallel** (R Core Team 2014) and **snow** (Tierney, Rossini, Li, and Sevcikova 2013).

Cumulative sums of $\log PL_t$ also allow for model comparison through cumulative log predictive Bayes factors. Letting $PL_t(A)$ denote the predictive likelihood of model A at time t, and $PL_t(B)$ the corresponding value of model B, the cumulative log predictive Bayes factor at time u (and starting point S) in favor of model A over model B is simply given through

$$\log \left[\frac{p_A(\boldsymbol{y}_{[S+1:u]}^o | \boldsymbol{y}_{[1:S]}^o)}{p_B(\boldsymbol{y}_{[S+1:u]}^o | \boldsymbol{y}_{[1:S]}^o)} \right] = \sum_{t=S+1}^{u} \log \left[\frac{PL_t(A)}{PL_t(B)} \right] = \sum_{t=S+1}^{u} [\log PL_t(A) - \log PL_t(B)]. \tag{9}$$

If the cumulative log predictive Bayes factor is positive at a given point in time, we have evidence in favor of model A, and vice versa. In this context, information up to time S is regarded as prior information, while out-of-sample predictive evaluation starts at time $S+1$. Note that the usual log Bayes factor is a special case of Equation 9 for $S = 0$ and $u = T$.

The top panel of Figure 16 gives an overview of the model performance through visual inspection of observed residuals against their predicted distributions. For the purpose of this paper, residuals are simply defined as the deviation from the median of the predicted distribution. It stands out that predictive quantiles arising from the model with SV errors exhibit much more flexibility to "adapt" to the "current state of the world", while the simpler homoskedastic model barely exhibits this feature. While there is little difference in predicted residuals until the beginning of 2007, the model with SV residuals performs substantially better during the pre-crisis era (less volatility) and during the actual crisis (more volatility). This picture is confirmed through the cumulative log predictive Bayes factors displayed in the bottom panel. Note that the last point plotted equals around 223, providing overwhelming overall evidence in favor of the model with SV errors, clearly rejecting the assumption of homoskedastic error terms.

For an example of using **stochvol** for univariate SV updates within a factor SV framework, see Kastner, Frühwirth-Schnatter, and Lopes (2014).

6. Conclusion

Aim of this paper was to introduce to the functionality of the R package **stochvol**, which provides a fully Bayesian simulation-based approach for inference in stochastic volatility models.

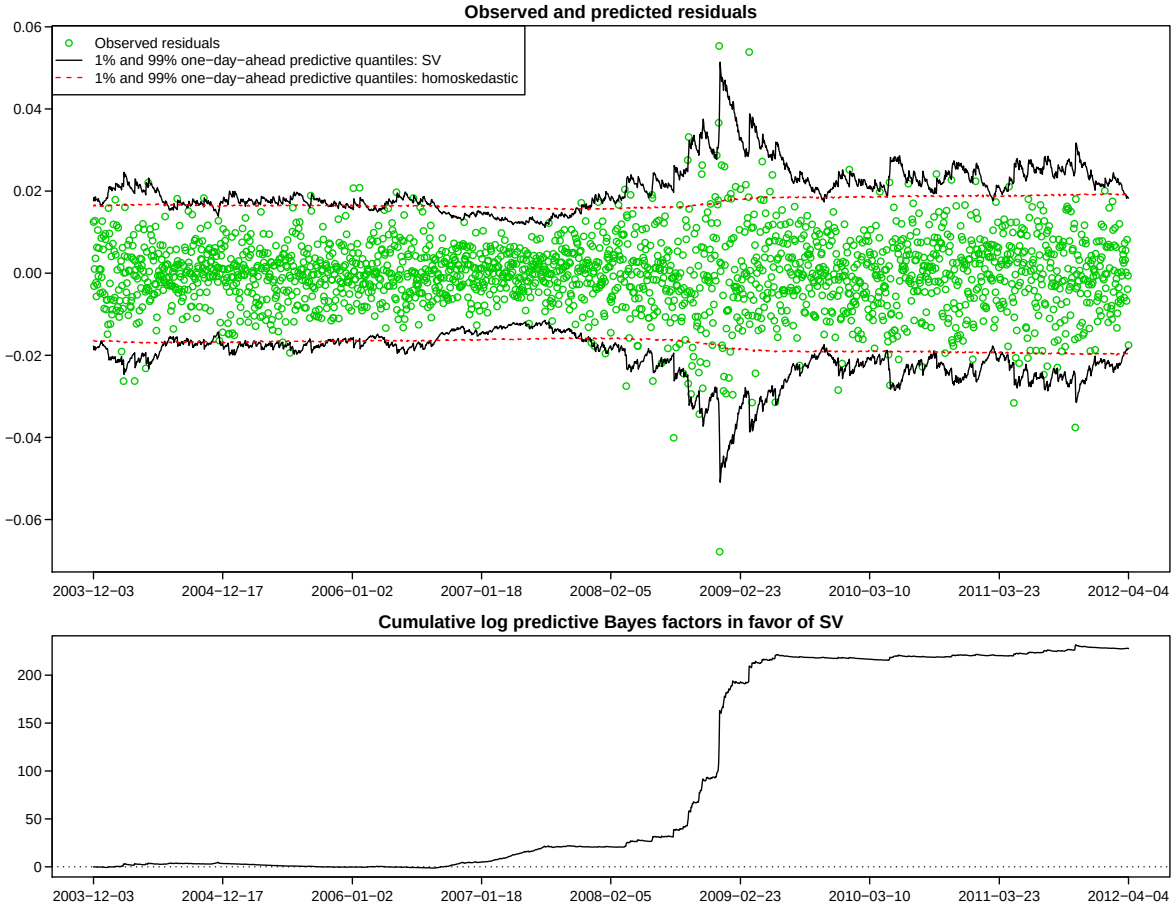

Figure 16: Top panel: Observed residuals with respect to the median of the one-day-ahead predictive distribution along with 1% and 99% quantiles of the respective predictive distributions. It can clearly be seen that the variance of the predictive distribution in the SV model adjusts to heteroskedasticity, while the model with homoskedastic errors is much more restrictive. Bottom panel: Cumulative log predictive Bayes factors in favor of the model with SV residuals. Values greater than zero mean that the model with SV residuals performs better out of sample up to this point in time.

The typical workflow occurring when using **stochvol** was presented by analyzing exchange rate data in the package's `exrates` data set. Furthermore, it was shown how the package can be used as a "plug-in" tool for other MCMC samplers. This was illustrated by estimating a Bayesian linear model with SV errors.

In the real-world example, raw exchange rates from EUR to USD were analyzed. Out-of-sample analysis through cumulative predictive Bayes factors clearly showed that already for a simple linear autoregressive model for the raw exchange rates, SV residuals substantially improve prediction performance, especially in turbulent times.

References

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

Bollerslev T (2008). "Glossary to ARCH (GARCH)." *Technical report*, CREATES Research Paper 2008-49. doi:10.2139/ssrn.1263250.

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.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. URL http://www.jstatsoft.org/v40/i08/.

Engle RF (1982). "Autoregressive Conditional Heteroscedasticity With Estimates of the Variance of United Kingdom Inflation." *Econometrica*, **50**(4), 987–1007. URL http://www.jstor.org/stable/1912773.

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.

Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2013). ***mvtnorm**: Multivariate Normal and t Distributions*. R package version 0.9-9996, URL http://CRAN.R-project.org/package=mvtnorm.

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.

Ghysels E, Harvey AC, Renault E (1996). "Stochastic Volatility." In GS Maddala, CR Rao (eds.), *Statistical Methods in Finance*, volume 14 of *Handbook of Statistics*, pp. 119–191. Elsevier. doi:10.1016/S0169-7161(96)14007-4.

Hills SE, Smith AFM (1992). "Parameterization Issues in Bayesian Inference." In JM Bernardo, JO Berger, AP Dawid, AFM Smith (eds.), *Proceedings of the Fourth Valencia International Meeting*, volume 4 of *Bayesian Statistics*, pp. 227–246. Oxford University Press.

Jacquier E, Polson NG, Rossi PE (1994). "Bayesian Analysis of Stochastic Volatility Models." *Journal of Business & Economic Statistics*, **12**(4), 371–389. 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.

Kastner G (2014). ***stochvol**: Efficient Bayesian Inference for Stochastic Volatility (SV) Models*. R package version 0.8-4, URL http://CRAN.R-project.org/package=stochvol.

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

Kastner G, Frühwirth-Schnatter S, Lopes HF (2014). "Analysis of Exchange Rates via Multivariate Bayesian Factor Stochastic Volatility Models." In E Lanzarone, I Francesca (eds.), *The Contribution of Young Researchers to Bayesian Statistics – Proceedings of BAYSM2013*, volume 63 of *Springer Proceedings in Mathematics & Statistics*, pp. 181–186. Springer.

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

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.

Meyer R, Yu J (2000). "BUGS for a Bayesian Analysis of Stochastic Volatility Models." *The Econometrics Journal*, **3**(2), 198–215. doi:10.1111/1368-423X.00046.

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.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

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.

Taylor SJ (1982). "Financial Returns Modelled by the Product of Two Stochastic Processes: A Study of Daily Sugar Prices 1691–79." In OD Anderson (ed.), *Time Series Analysis: Theory and Practice 1*, pp. 203–226. North-Holland, Amsterdam.

Tierney L, Rossini AJ, Li N, Sevcikova H (2013). ***snow**: Simple Network of Workstations*. R package version 0.3-12, URL http://CRAN.R-project.org/package=snow.

Yu J (2005). "On Leverage in a Stochastic Volatility Model." *Journal of Econometrics*, **127**(2), 165–178. doi:10.1016/j.jeconom.2004.08.002.

Yu Y, Meng XL (2011). "To Center or Not to Center: That is Not the Question—An Ancillarity-Suffiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency." *Journal of Computational and Graphical Statistics*, **20**(3), 531–570. doi:10.1198/jcgs.2011.203main.

Affiliation:

Gregor Kastner
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
WU Vienna University of Economics and Business
Welthandelsplatz 1
1020 Vienna, Austria
E-mail: gregor.kastner@wu.ac.at