MDSV: An R package for estimating and forecasting financial data with MDSV model

Kassimou Abdoul Haki Maoude

2024-02-10

1 Introduction

Regime-switching processes are popular tools to interpret, model and forecast financial data. The Markov-switching multifractal (MSM) model of Calvet and Fisher (2004) has proved to be a strong competitor to the GARCH class of models for modeling the volatility of returns. In this model, volatility dynamics are driven by a latent high-dimensional Markov chain constructed by multiplying independent two-state Markov chains. We propose the multifractal discrete stochastic volatility (MDSV) model as a generalization of the MSM process and of other related high-dimensional hidden Markov models (Fleming and Kirby, 2013; Cordis and Kirby, 2014; Augustyniak et al., 2019). Our model is intended to model financial returns and realized volatilities jointly or uniquely, and therefore also extends existing high-dimensional Markov-switching processes to the joint setting. Our approach consists in building a highdimensional Markov chain by the product of lower-dimensional Markov chains which have a discrete stochastic volatility representation.

We also present an easy-to-use R package—named **MDSV**—for implementing the model. This note is a supplement for the package manual. We provide examples with data from Oxford-Man institute. The package is located on GitHub at github.com/Abdoulhaki/MDSV. All the results of the note can be replicated following the code provided.

2 Model

Let r_t and RV_t denote, respectively, the demeaned log-return and realized variance of a financial asset from time $t-1$ to t, for $t = 1, \ldots, T$. Our proposed MDSV model postulates that the univariate serie $\{r_t\}$ or $\{RV_t\}$ or the joint time series $\{(r_t, RV_t)\}$ is driven by a MDSV process denoted by $\{V_t\}$. This process is a latent variance process constructed from the product of a high-dimensional Markov chain $\{C_t\}$ governing volatility persistence and of a data-driven component $\{L_t\}$ capturing the leverage effect, that is

$$V_t = C_t L_t.$$

To relate the univariate serie r_t to this process, we assume that

$$r_t = \sqrt{V_t}\epsilon_t \,, \tag{1}$$

where $\{\epsilon_t\}$ is a serially independent normal innovation processes with mean 0 and variance 1. For the univariate serie RV_t, we assume that

$$RV_t = V_t \eta_t \,, \tag{2}$$

where $\{\eta_t\}$ is a serially independent gamma innovation processes with mean 1 (shape γ and scale $1/\gamma$. In the case of the joint framework, to relate (r_t, RV_t) to the latent process, we assume that

$$r_t = \sqrt{V_t}\epsilon_t, \tag{3}$$

$$\log RV_t = \xi + \varphi \log V_t + \delta_1 \epsilon_t + \delta_2 \left(\epsilon_t^2 - 1\right) + \gamma \varepsilon_t, \tag{4}$$

where $\xi \in \mathbb{R}$, $\varphi \in \mathbb{R}$, $\delta_1 \in \mathbb{R}$, $\delta_2 \in \mathbb{R}$ and $\gamma \in (0,\infty)$ are parameters, and $\{\epsilon_t\}$ and $\{\varepsilon_t\}$ are mutually and serially independent normal innovation processes with mean 0 and variance 1.

2.1 Volatility persistance component

The Volatility persistance component $\{C_t\}$ is constructed from the product of N independent Markov chains with dimension K, denoted by $\left\{C_t^{(i)}\right\}$, $i = 1\ldots,N$, that is,

$$C_t = \frac{\sigma^2}{c_0}\prod_{i=1}^{N} C_t^{(i)}, \tag{5}$$

where $\sigma \in (0,\infty)$ is a parameter and the constant $c_0 = \mathbb{E}\left[\prod_{i=1}^N C_t^{(i)}\right] = \prod_{i=1}^N \mathbb{E}\left[C_t^{(i)}\right]$ is defined such that $\sigma^2 = \mathbb{E}[C_t]$.

Each component $\left\{C_t^{(i)}\right\}$, $i = 1,\ldots,N$ is a Markov chain with $K \times K$ transition matrix $\mathbf{P}^{(i)}$ defined by

$$\mathbf{P}^{(i)} = \phi^{(i)}\mathbf{I}_K + (1-\phi^{(i)})\mathbf{1}_K\boldsymbol{\pi}^{(i)\prime}, \tag{6}$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix, $\mathbf{1}_K$ is a vector of size K composed of ones, $\phi^{(i)} \in [0,1)$ is a parameter, and $\boldsymbol{\pi}^{(i)}$ is a parameter vector of probabilities corresponding to the stationary distribution of $\left\{C_t^{(i)}\right\}$. The state space of $\left\{C_t^{(i)}\right\}$ is denoted by the parameter vector $\boldsymbol{\nu}^{(i)}$.

For tractability and to avoid over-parametrization, we further impose the following constraints on the model parameters, for $i = 1,\ldots,N$, and $j = 1,\ldots,K$,

$$\begin{aligned}
\phi^{(i)} &= a^{b^{i-1}}, \\
\boldsymbol{\nu}^{(i)} &= \boldsymbol{\nu}^{(1)} = \begin{pmatrix}\nu_1^{(1)} & \nu_2^{(1)} & \ldots & \nu_K^{(1)}\end{pmatrix}' \\
\nu_j^{(1)} &= \nu_0\left(\frac{2-\nu_0}{\nu_0}\right)^{j-1}, \\
\boldsymbol{\pi}^{(i)} &= \boldsymbol{\pi}^{(1)} = (\pi_1^{(1)},\ldots,\pi_K^{(1)})', \\
\pi_j &= \binom{K-1}{j-1}\omega^{j-1}(1-\omega)^{K-j},
\end{aligned} \tag{7}$$

in which $\nu_0 \in (0,1)$, $\omega \in (0,1)$, $a \in (0,1)$ and $b \in [1,+\infty)$.

We write MDSV(N,K) to designate the MDSV model with a Markov chain $\{C_t\}$ constructed from the product of N Markov chains with dimension K. The state space of $\{C_t\}$, denoted by $\boldsymbol{\nu}$, has dimension K^N and is given by

$$\boldsymbol{\nu} = \frac{\sigma^2}{c_0}\left[\otimes_{i=1}^N\left(\boldsymbol{\nu}^{(i)}\right)\right] = \frac{\sigma^2}{c_0}\left(\boldsymbol{\nu}^{(1)}\right)^{\otimes N}.$$

The transition matrix of $\{C_t\}$, denoted by $\mathbf{P}$, is given by

$$\mathbf{P} = \otimes_{i=1}^N\left(\mathbf{P}^{(i)}\right).$$

Finally, the stationary distribution of $\{C_t\}$, denoted by $\boldsymbol{\pi}$, is given by

$$\boldsymbol{\pi} = \otimes_{i=1}^N\left(\boldsymbol{\pi}^{(i)}\right) = \left(\boldsymbol{\pi}^{(1)}\right)^{\otimes N}.$$

2.2 Leverage effect component

A process $\{L_t\}$ to capture a time-varying leverage effect is add in the latent volatility process $\{V_t\}$. This approach of capturing leverage effect is a very perfromant way introduced by [1]. The leverage effect is defined as :

$$L_t = \prod_{i=1}^{N_L} \left(1 + l_i \frac{|r_{t-i}|}{\sqrt{L_{t-i}}} \mathbf{1}_{\{r_{t-i}<0\}} \right) , \quad \text{where} \quad l_i = \theta_l^{i-1} l_1 \text{ and } l_1 > 0 \, , \theta_l \in [0, 1] \, .$$

This specfication of the leverage process is give the propriety to this component to be a predictable process as for each t, it value is fully determined by the N_L previous obseved log-return (up to the date $t-1$).

Moreover, this specification has a nice interpretation. In fact a negative past log-return add an additional volatility of intensity related to magnitude of log-return and a parameter l_i structured such as to give less and less importance to the most distant log-returns.

3 Presentation of the package

In this section, we present each function of the package and some examples to show how to use it. The package MDSV can be loaded as a common package in R.

```
library(MDSV)
```

The parameters that specify a model in the contexte of MDSV package are :

- `N` : The number of components for the MDSV process.
- `K` : The number of states of each MDSV process component.
- `ModelType` : An integer designing the type of model to be fit. `0` for univariate log-returns, `1` for univariate realized variances and `2` for joint log-return and realized variances.
- `LEVIER` : A logical designing if the MDSV model take leverage effect into account or not.

3.1 Fitting

The `MDSVfit` method fit the MDSV model on log-retruns and realized variances (uniquely or jointly). It takes the following arguments:

```
args(MDSVfit)
```

```
## function (N, K, data, ModelType = 0, LEVIER = FALSE, start.pars = list(),
##     ...)
## NULL
```

The MDSV optimization routine set of feasible starting points which are used to initiate the MDSV recursion. The likelihood calculation is performed in C++ through the `Rcpp` package. The optimization is perform using the `solnp` solver of the `Rsolnp` package and additional options can be supply to the fonction. While fitting an univariate realized variances data, log-returns are required to add leverage effect. Information criterias AIC and BIC are computed using the formulas :

- $AIC = \mathcal{L} - k \, ,$
- $BIC = \mathcal{L} - (k/2) * log(T) \, ,$

where $\mathcal{L}$ is the log-likelihood, k is the number of parameters and T the number of observations in the dataset. The fitted object is of class `MDSVfit` which can be passed to a variety of other methods such as summary, plot, MDSVboot. The following examples illustrate its use, but the interested reader should consult the documentation on the methods available for the returned class.

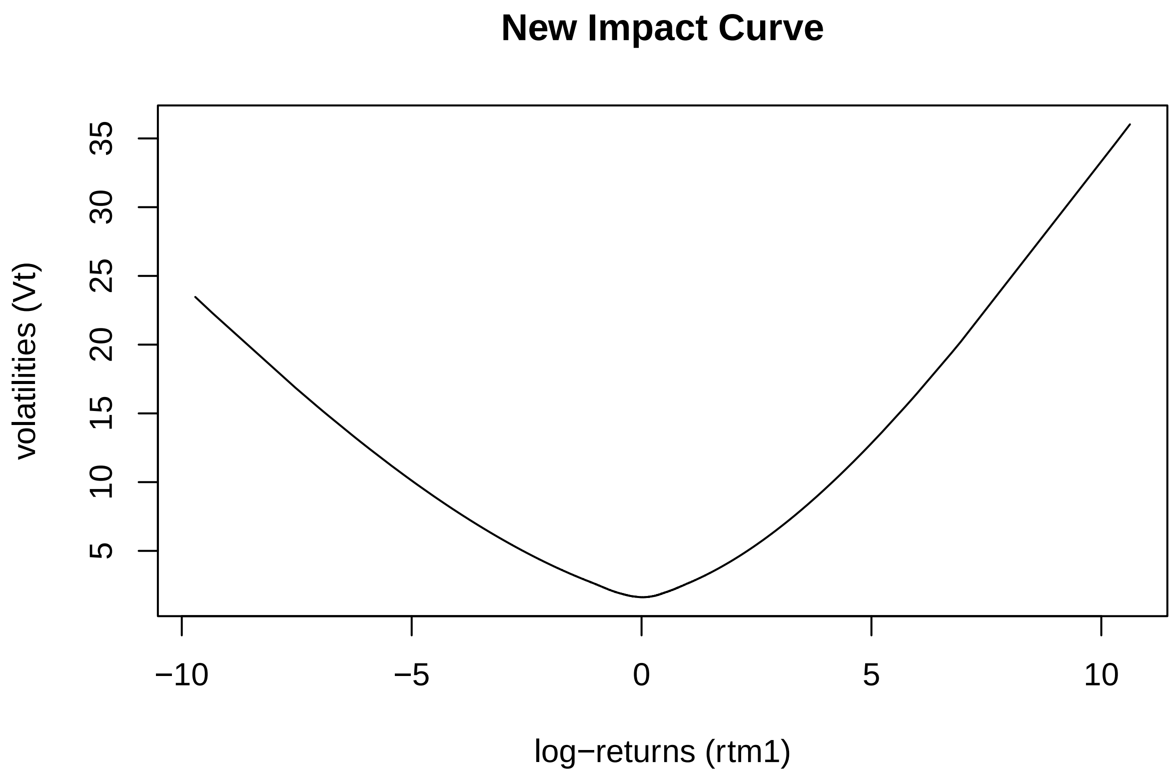
[1] Augustyniak et al. (2019)

```r
start.date <- as.Date("2000-01-03")
end.date   <- as.Date("2019-08-31")
data(sp500)
sp500      <- sp500[rownames(sp500)>=start.date & rownames(sp500)<=end.date,]
N          <- 2
K          <- 3
LEVIER     <- TRUE

# Model estimation : univariate log-returns (ModelType = 0)
out        <- MDSVfit(K = K, N = N, data = sp500, ModelType = 0, LEVIER = LEVIER)
# Summary
summary(out)
```

```
## =================================================
## =================  MDSV fitting =================
## =================================================
##
## Model   : MDSV(2,3)
## Data    : Univariate log-return
## Leverage: TRUE
##
## Optimal Parameters
## ------------------------------------------------
## Convergence : Convergence.
## omega    : 0.358702
## a    : 0.998043
## b    : 14.041846
## sigma    : 0.271698
## v0   : 0.697954
## l    : 0.753797
## theta    : 0.914732
##
## LogLikelihood : -6537.58
##
## Information Criteria
## ------------------------------------------------
## AIC  : -6544.58
## BIC  : -6567.35
```

```r
# Plot
plot(out,c("nic"))
```

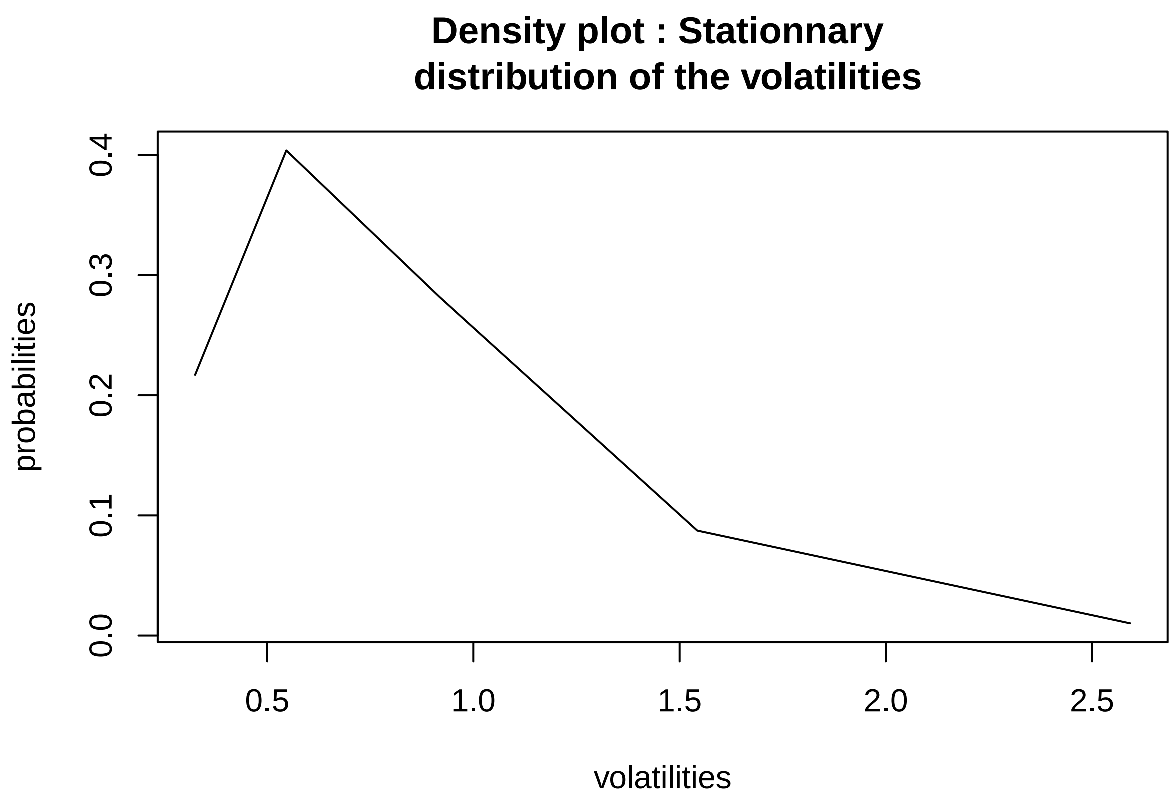



```r
# Model estimation : univariate realized variances (ModelType = 1) without leverage
out       <- MDSVfit(K = K, N = N, data = sp500, ModelType = 1, LEVIER = FALSE)
# Summary
summary(out)
```

```
## ==================================================
## =================  MDSV fitting =================
## ==================================================
##
## Model   : MDSV(2,3)
## Data    : Univariate realized variances
## Leverage: FALSE
##
## Optimal Parameters
## --------------------------------------------------
## Convergence : Convergence.
## omega    : 0.317485
## a    : 0.989743
## b    : 12.039114
## sigma    : 0.657176
## v0   : 0.523017
## shape    : 3.893613
##
## LogLikelihood : -1481.25
##
## Information Criteria
```

```
## --------------------------------------------------
## AIC  : -1487.25
## BIC  : -1506.76
```

```r
# Plot
plot(out,c("dis"))
```

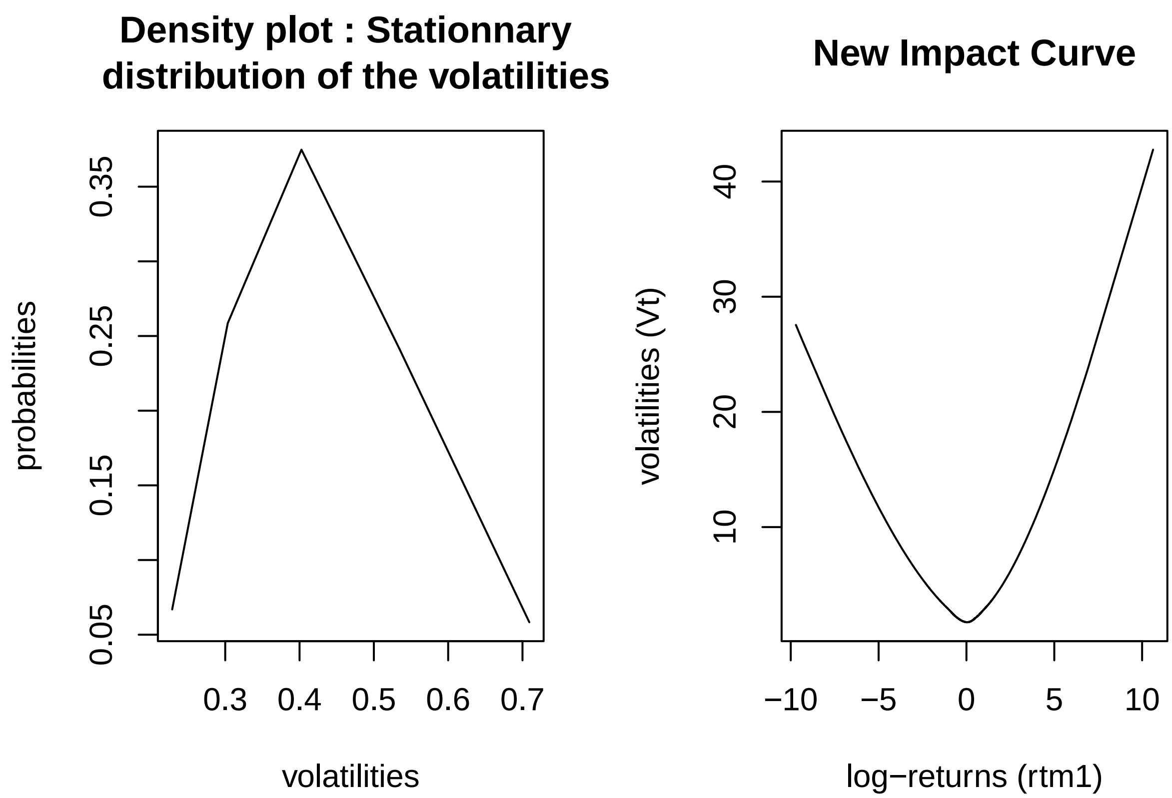



```r
# Model estimation : joint model (ModelType = 2)
out       <- MDSVfit(K = K, N = N, data = sp500, ModelType = 2, LEVIER = LEVIER)
# Summary
summary(out)
```

```
## ==================================================
## =================  MDSV fitting =================
## ==================================================
##
## Model   : MDSV(2,3)
## Data    : Joint log-return and realized variances
## Leverage: TRUE
##
## Optimal Parameters
## --------------------------------------------------
## Convergence : Convergence.
## omega    : 0.491414
## a    : 0.931017
## b    : 1.000047
## sigma    : 0.18632
```

```
## v0   : 0.724304
## xi   : -0.412812
## varphi   : 0.98649
## delta1   : -0.110335
## delta2   : 0.103976
## shape    : 0.192582
## l    : 0.832373
## theta    : 0.922564
##
## LogLikelihood : -6777.87
##
## Information Criteria
## ------------------------------------------------
## AIC  : -6789.87
## BIC  : -6828.9
```

```r
# Plot
plot(out,c("dis","nic"))
```



3.2 Filtering

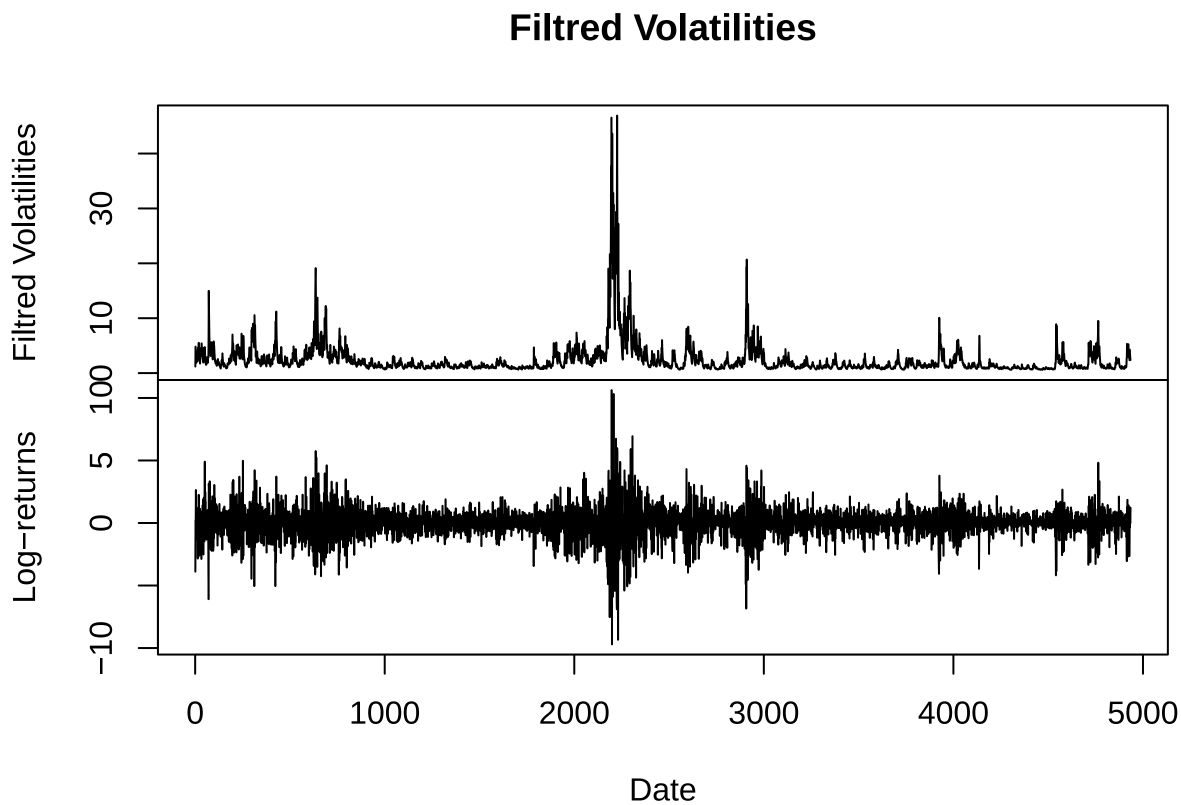
Sometimes it is desirable to simply filter a set of data with a predefined set of parameters. This may for example be the case when new data has arrived and one might not wish to re-fit. The `MDSVfilter` method does exactly that, filter the MDSV model on log-retruns and realized variances (uniquely or jointly) data with a predefined set of parameters. The examples which follow explain how:

```r
N          <- 3
K          <- 3
LEVIER     <- TRUE
para       <- c(omega = 0.52, a = 0.99, b = 2.77, sigma = 1.95, v0 = 0.72,
               l = 0.78, theta = 0.876)

# Model filtering : univariate log-returns (ModelType = 0)
out        <- MDSVfilter(K = K, N = N,data = sp500, para = para, ModelType = 0, LEVIER = LEVIER, calcula
# Summary
summary(out)
```

```
## =================================================
## ================  MDSV Filtering ================
## =================================================
##
## Model   : MDSV(3,3)
## Data    : Univariate log-return
## Leverage: TRUE
##
## Optimal Parameters
## -------------------------------------------------
## omega    : 0.52
## a    : 0.99
## b    : 2.77
## sigma    : 1.95
## v0   : 0.72
## l    : 0.78
## theta    : 0.876
##
## LogLikelihood    : -6867.3
##
## Information Criteria
## -------------------------------------------------
## AIC  : -6874.3
## BIC  : -6897.06
##
## Value at Risk
## -------------------------------------------------
## 95%      : -5.3983161448555
```

```r
# Plot
plot(out)
```




```r
para      <- c(omega = 0.52, a = 0.99, b = 2.77, sigma = 1.95, v0 = 0.72, shape = 2.10)

# Model filtering : univariate realized variances (ModelType = 1) without leverage
out       <- MDSVfilter(K = K, N = N, data = sp500, para = para, ModelType = 1, LEVIER = FALSE, calcula
```

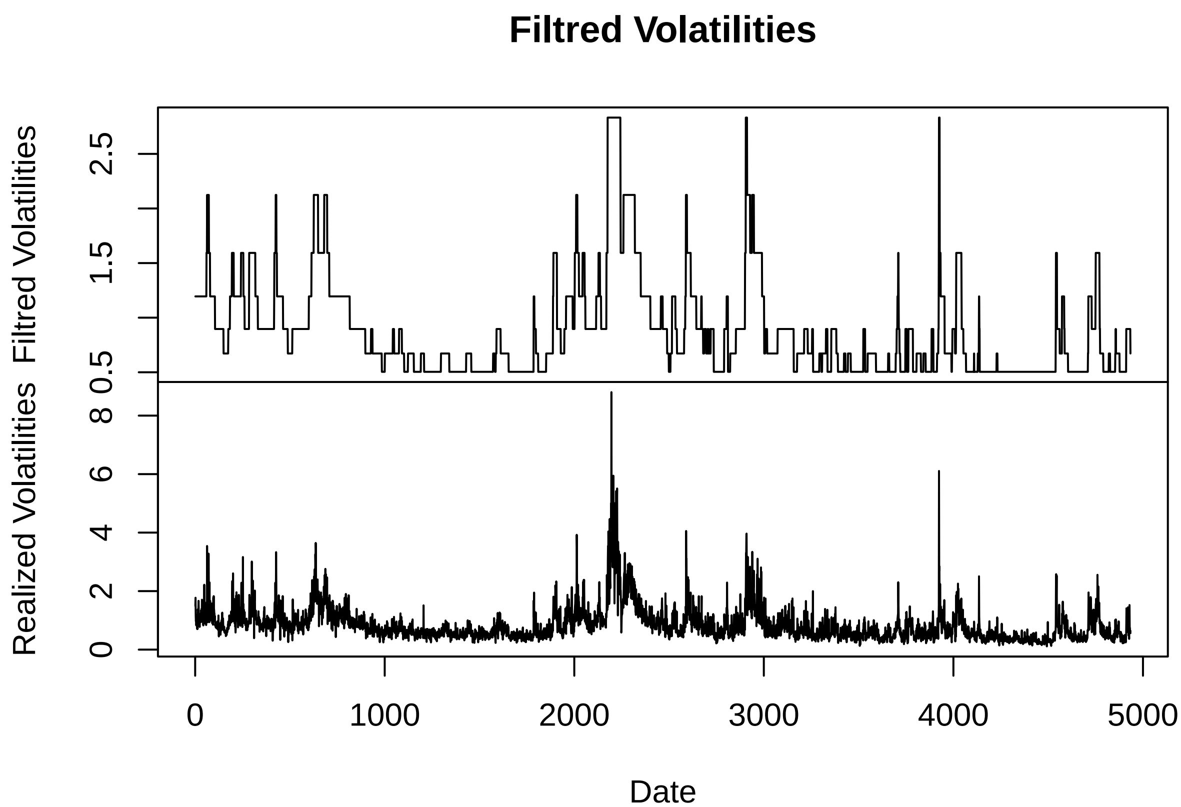
```
## [1] "MDSVfilter() WARNING: VaR are compute only for log-returns! calculate.VaR set to FALSE!"
```

```r
# Summary
summary(out)
```

```
## =================================================
## ================  MDSV Filtering ================
## =================================================
##
## Model   : MDSV(3,3)
## Data    : Univariate realized variances
## Leverage: FALSE
##
## Optimal Parameters
## -------------------------------------------------
## omega    : 0.52
## a    : 0.99
## b    : 2.77
## sigma    : 1.95
## v0   : 0.72
## shape    : 2.1
##
```

```
## LogLikelihood    : -1921.72
##
## Information Criteria
## ------------------------------------------------
## AIC  : -1927.72
## BIC  : -1947.23
```

```
# Plot
plot(out)
```



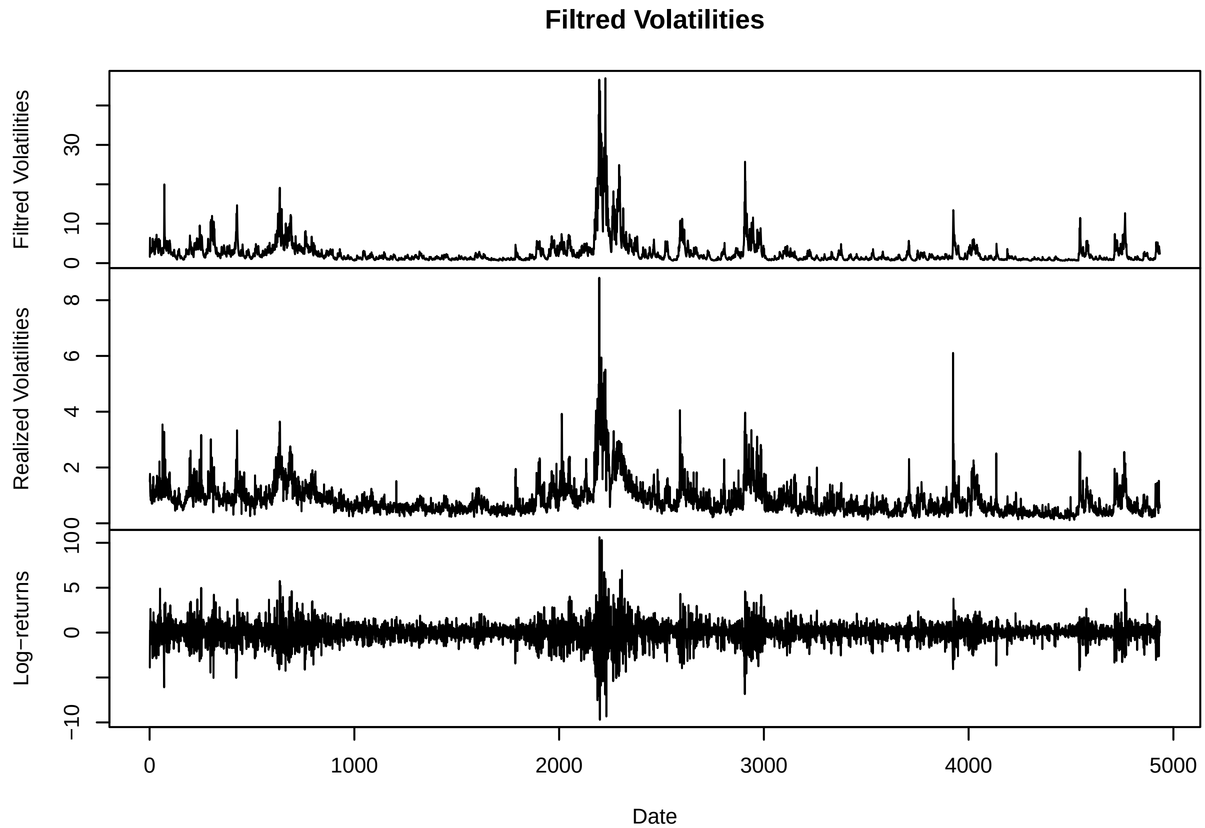

```
para       <- c(omega = 0.52, a = 0.99, b = 2.77, sigma = 1.95, v0 = 0.72,
                xi = -0.5, varphi = 0.93, delta1 = 0.93, delta2 = 0.04, shape = 2.10,
                l = 0.78, theta = 0.876)

# Model filtering : joint model (ModelType = 2)
out        <- MDSVfilter(K = K, N = N,data = sp500, para = para, ModelType = 2, LEVIER = LEVIER, calcula
# Summary
summary(out)
```

```
## ==================================================
## ================  MDSV Filtering ================
## ==================================================
##
## Model   : MDSV(3,3)
## Data    : Joint log-return and realized variances
## Leverage: TRUE
##
```

```
## Optimal Parameters
## -------------------------------------------------
## omega    : 0.52
## a    : 0.99
## b    : 2.77
## sigma    : 1.95
## v0   : 0.72
## xi   : -0.5
## varphi   : 0.93
## delta1   : 0.93
## delta2   : 0.04
## shape    : 2.1
## l    : 0.78
## theta    : 0.876
##
## LogLikelihood    : -11411.8
## Marginal LogLikelihood : -6659
##
## Information Criteria
## -------------------------------------------------
## AIC  : -11423.8
## BIC  : -11462.82
##
## Value at Risk
## -------------------------------------------------
## 95%      : -5.871222257462
```

```r
# Plot
plot(out)
```



The returned object is of class `uGARCHfilter` and shares many of the methods as the uGARCHfit class. Additional arguments to the function are explained in the documentation.

3.3 Forecasting and the MDSV Bootstrap

When the MDSV model does not take leverage effect into account, forecasting techniques developed by Hamilton (1994) are applicable in MDSV framework. But, when the MDSV model takes leverage effect into account, it is not possible to have analytic formula for $h > 1$ ahead forecasts. Thoses forecasts are then performed through bootstrap simulations. The following examples provides for a brief look at the `MDSVboot` method, but the interested reader should consult the more comprehensive examples in the inst folder of the package.

```
N         <- 3
K         <- 3
LEVIER    <- TRUE

# Model forecasting : univariate log-returns (ModelType = 0) without leverage
out_fit   <- MDSVfit(K = K, N = N, data = sp500, ModelType = 0, LEVIER = FALSE)
out       <- MDSVboot(fit = out_fit, n.ahead = 100, n.bootpred = 10000, rseed = 349)
# Summary
summary(out)
```

```
## =================================================
## ==========  MDSV Bootstrap Forecasting ==========
## =================================================
##
## Model       : MDSV(3,3)
```

```
## Data        : Univariate log-return
## Leverage    : FALSE
## n.ahead     : 100
## Date (T[0]) : 4934
##
## Sigma (summary) :
##      t+1      t+2      t+3      t+4      t+5      t+6      t+7      t+8
## 1.275306 1.277392 1.279191 1.280721 1.282001 1.283047 1.283874 1.284496
##      t+9     t+10
## 1.284928 1.285182
## ........................
```

```
# Model forecasting : univariate realized variances (ModelType = 1) without leverage
out_fit   <- MDSVfit(K = K, N = N, data = sp500, ModelType = 1, LEVIER = FALSE)
out       <- MDSVboot(fit = out_fit, n.ahead = 100, n.bootpred = 10000, rseed = 349)
# Summary
summary(out)
```

```
## =================================================
## ==========  MDSV Bootstrap Forecasting ==========
## =================================================
##
## Model       : MDSV(3,3)
## Data        : Univariate realized variances
## Leverage    : FALSE
## n.ahead     : 100
## Date (T[0]) : 4934
##
## Realized Variances (summary) :
##      t+1      t+2      t+3      t+4      t+5      t+6      t+7      t+8
## 0.433671 0.488964 0.530670 0.561927 0.585149 0.602200 0.614514 0.623197
##      t+9     t+10
## 0.629102 0.632886
## ........................
```

```
# Model bootstrap forecasting : joint model (ModelType = 2) with leverage
out_fit   <- MDSVfit(K = K, N = N, data = sp500, ModelType = 2, LEVIER = LEVIER)
out       <- MDSVboot(fit = out_fit, n.ahead = 100, n.bootpred = 10000, rseed = 349)
# Summary
summary(out)
```

```
## =================================================
## ==========  MDSV Bootstrap Forecasting ==========
## =================================================
##
## Model       : MDSV(3,3)
## Data        : Joint log-return and realized variances
## Leverage    : TRUE
## n.ahead     : 100
## Date (T[0]) : 4934
##
## Log-returns (summary) :
##            min      q.25      mean    median      q.75      max
## t+1  -2.813077 -0.478486 -0.001308  0.002281 0.481019 2.954423
## t+2  -3.759899 -0.483744 -0.005290  0.002751 0.477142 4.246142
```

```
## t+3  -4.938748 -0.457942  0.001235 -0.002173 0.462083 2.894166
## t+4  -3.587506 -0.455700 -0.001500  0.004455 0.460698 4.296521
## t+5  -3.790773 -0.446988 -0.000615 -0.001967 0.446722 4.425511
## t+6  -4.275248 -0.447481  0.004432  0.000327 0.446989 5.114404
## t+7  -6.330208 -0.440789 -0.000528  0.003860 0.448376 4.314299
## t+8  -6.888472 -0.429485 -0.004297  0.000019 0.419219 4.653410
## t+9  -4.331712 -0.422363  0.008948  0.014006 0.439367 6.616273
## t+10 -4.641084 -0.442790 -0.010674 -0.007501 0.416302 5.236948
## ........................
##
## Realized Variances (summary) :
##      min     q.25     mean   median     q.75       max
## t+1    0 0.041753 0.380788 0.177263 0.503056  6.085136
## t+2    0 0.040676 0.385201 0.177926 0.492797 12.320417
## t+3    0 0.039269 0.378870 0.163694 0.479723 16.529154
## t+4    0 0.035878 0.395575 0.162754 0.487714 12.606303
## t+5    0 0.034084 0.390952 0.154480 0.466580 13.352795
## t+6    0 0.034081 0.390917 0.154875 0.454031 17.691705
## t+7    0 0.035516 0.398121 0.153147 0.449671 26.785938
## t+8    0 0.030907 0.386020 0.138969 0.432863 31.571275
## t+9    0 0.034081 0.390500 0.144140 0.444874 29.190327
## t+10   0 0.030588 0.391590 0.142543 0.440474 18.525459
## ........................
```

3.4 Simulation

The `MDSVsim` method takes the following arguments:

```r
args(MDSVsim)
```

```
## function (N, K, para, ModelType = 0, LEVIER = FALSE, n.sim = 1000,
##     n.start = 0, m.sim = 1, rseed = NA)
## NULL
```

where the `n.sim` indicates the length of the simulation while `m.sim` the number of independent simulations. Key to replicating results is the `rseed` argument which is used to pass a user seed to initialize the random number generator, else one will be assigned by the program.

3.5 Rolling Estimation

The `MDSVroll` method allows to perform a rolling estimation and forecasting of a model/dataset combination, optionally returning the VaR at specified levels. More importantly, the `MDSVroll` method present the forecasting performance of the model by computing the RMSE, MAE and QLIK loss functions (see Patton, 2011). The following example illustrates the use of the method where use is also made of the parallel functionality and run on 7 cores. The `MDSVroll` object returned can be passed to the plot function. Additional methods, and more importantly extractor functions can be found in the documentation. As the `MDSVroll` method could take a certain time to execute, the package perform a progression bar to inform about the evolution.

```r
N               <- 2
K               <- 3
ModelType       <- 2
LEVIER          <- FALSE
n.ahead         <- 100
forecast.length <- 756
refit.every     <- 63
```

```r
refit.window     <- "recursive"
calculate.VaR    <- TRUE
VaR.alpha        <- c(0.01, 0.05, 0.1)
cluster          <- parallel::makeCluster(7)
rseed            <- 125

# rolling forecasts
out<-MDSVroll(N=N, K=K, data=sp500, ModelType=ModelType, LEVIER=LEVIER, n.ahead = n.ahead,
              forecast.length = forecast.length, refit.every = refit.every,
              refit.window = refit.window, window.size=NULL,calculate.VaR = calculate.VaR,
              VaR.alpha = VaR.alpha, cluster = cluster, rseed = rseed)
```

```
## Estimation step 1:
##   |                                                                              |
## Estimation step 2:
##   |                                                                              |
```

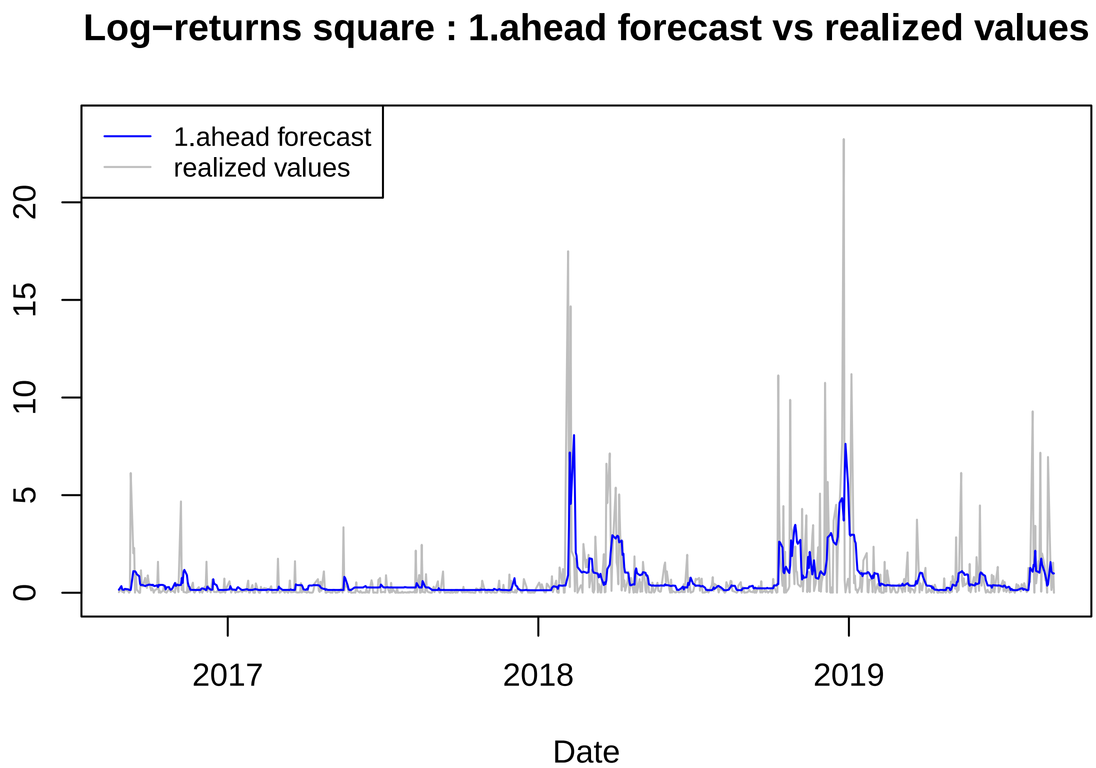
```r
parallel::stopCluster(cluster)
# Summary
summary(out, VaR.test=TRUE, Loss.horizon = c(1,5,10,25,50,75,100), Loss.window = 756)
```

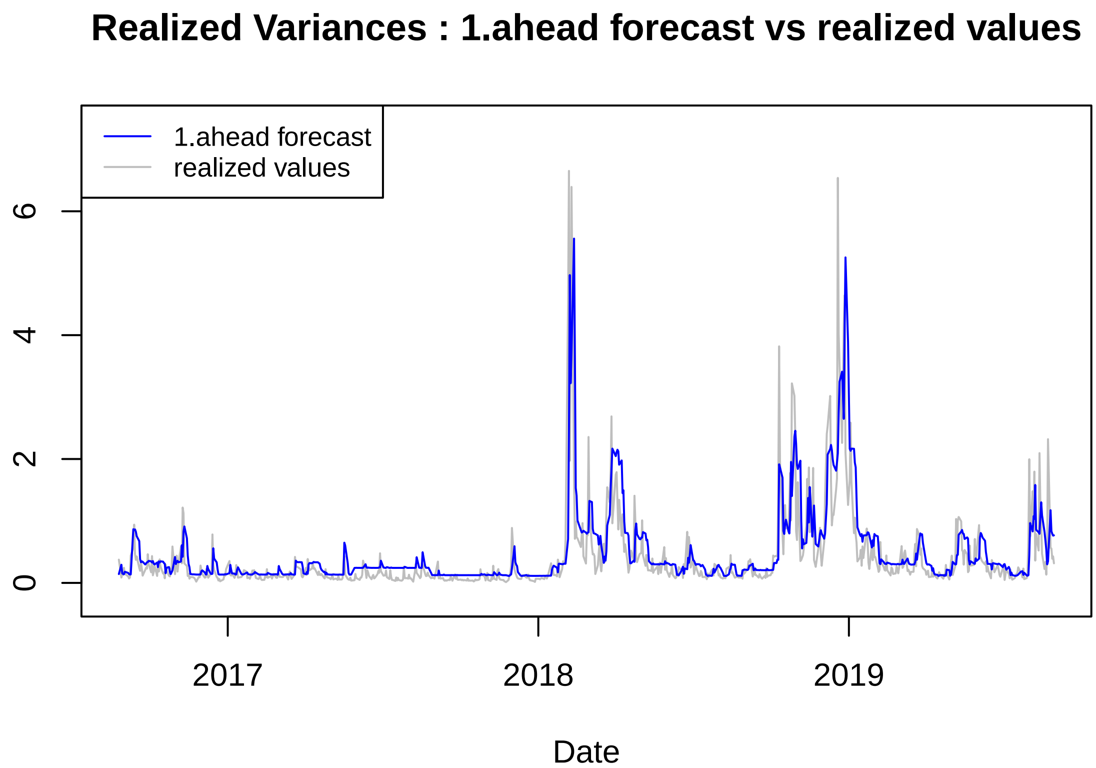
```
## ==================================================
## ===  MDSV Rolling Estimation and Forecasting ====
## ==================================================
##
## Model                : MDSV(2,3)
## Data                 : Joint log-return and realized variances
## Leverage             : FALSE
## No.refit             : 12
## Refit Horizon        : 63
## No.Forecasts         : 756
## n.ahead              : 100
## Date (T[0])          : 2016-08-25
##
## Forecasting performances
## --------------------------------------------------
## Predictive density : -735.38
## --------------------
##
## Cummulative Loss Functions :
## ----------------------------
## Log-returns :
##          1     5    10    25    50    75   100
## QLIK 0.192 1.919 2.682 3.721 4.511 5.012 5.325
## RMSE 1.668 0.943 0.832 0.753 0.747 0.783 0.764
## MAE  0.691 0.481 0.485 0.543 0.613 0.660 0.663
##
## Realized Variances :
##           1     5    10    25    50    75   100
## QLIK -0.318 1.435 2.222 3.287 4.093 4.587 4.908
## RMSE  0.511 0.492 0.495 0.506 0.527 0.558 0.558
## MAE   0.247 0.275 0.311 0.391 0.454 0.481 0.486
```

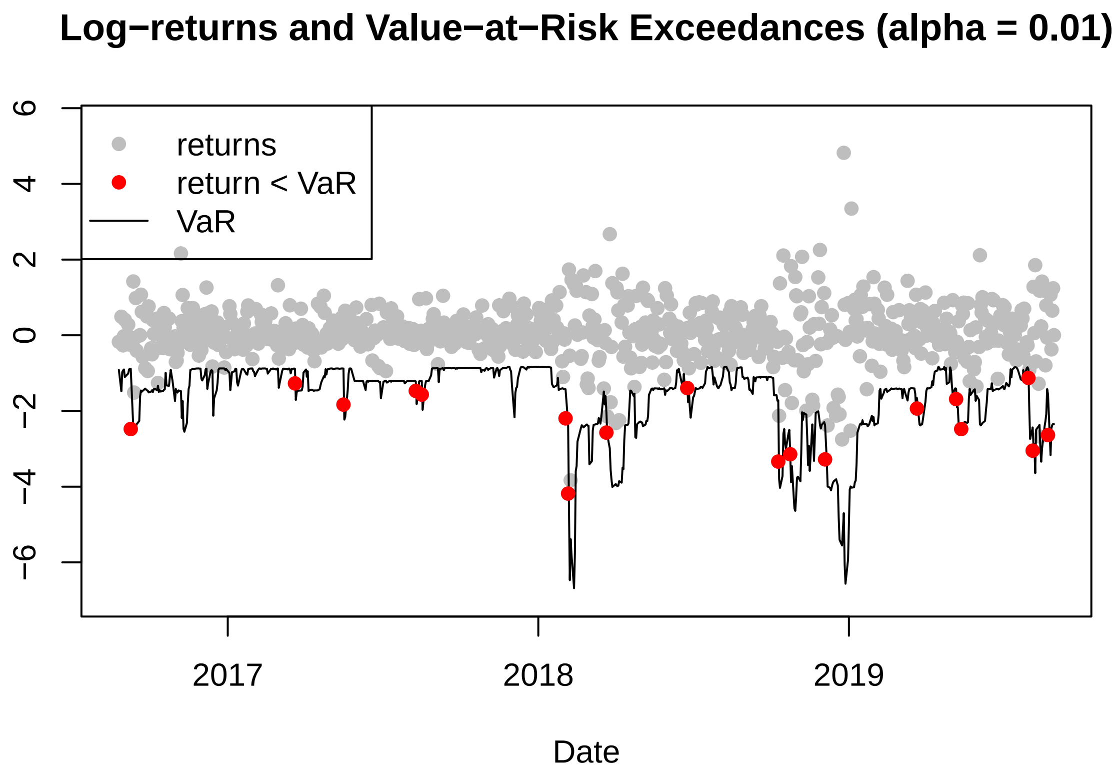
```
##
## Marginal Loss Functions :
## -------------------------
## Log-returns :
##          1     5    10    25    50    75   100
## QLIK 0.192 0.416 0.496 0.680 0.813 0.822 0.699
## RMSE 1.668 0.347 0.177 0.073 0.038 0.026 0.019
## MAE  0.691 0.156 0.086 0.039 0.022 0.015 0.012
##
## Realized Variances :
##           1      5    10    25    50    75   100
## QLIK -0.318 -0.071 0.053 0.242 0.367 0.389 0.316
## RMSE  0.511  0.140 0.073 0.033 0.018 0.012 0.009
## MAE   0.247  0.077 0.045 0.023 0.013 0.009 0.007
##
## VaR Tests
## ------------------------------------------------
## alpha              : 0.01%
## Excepted Exceed    : 7.6
## Actual VaR Exceed  : 18
## Actual %           : 0.02%
##
## Unconditionnal Coverage (Kupiec)
## Null-Hypothesis    : Correct exceedances
## LR.uc Statistic    : 10.496
## LR.uc Critical     : 6.635
## LR.uc p-value      : 0.001
## Reject Null        : Yes
##
## Independance (Christoffersen)
## Null-Hypothesis    : Independance of failures
## LR.ind Statistic   : 0.59
## LR.ind Critical    : 6.635
## LR.ind p-value     : 0.443
## Reject Null        : No
##
## Conditionnal Coverage (Christoffersen)
## Null-Hypothesis    : Correct exceedances and Independance of failures
## LR.cc Statistic    : 11.086
## LR.cc Critical     : 9.21
## LR.cc p-value      : 0.004
## Reject Null        : Yes
##
## ------------------------------------------------
## alpha              : 0.05%
## Excepted Exceed    : 37.8
## Actual VaR Exceed  : 42
## Actual %           : 0.06%
##
## Unconditionnal Coverage (Kupiec)
## Null-Hypothesis    : Correct exceedances
## LR.uc Statistic    : 0.475
## LR.uc Critical     : 3.841
## LR.uc p-value      : 0.491
```

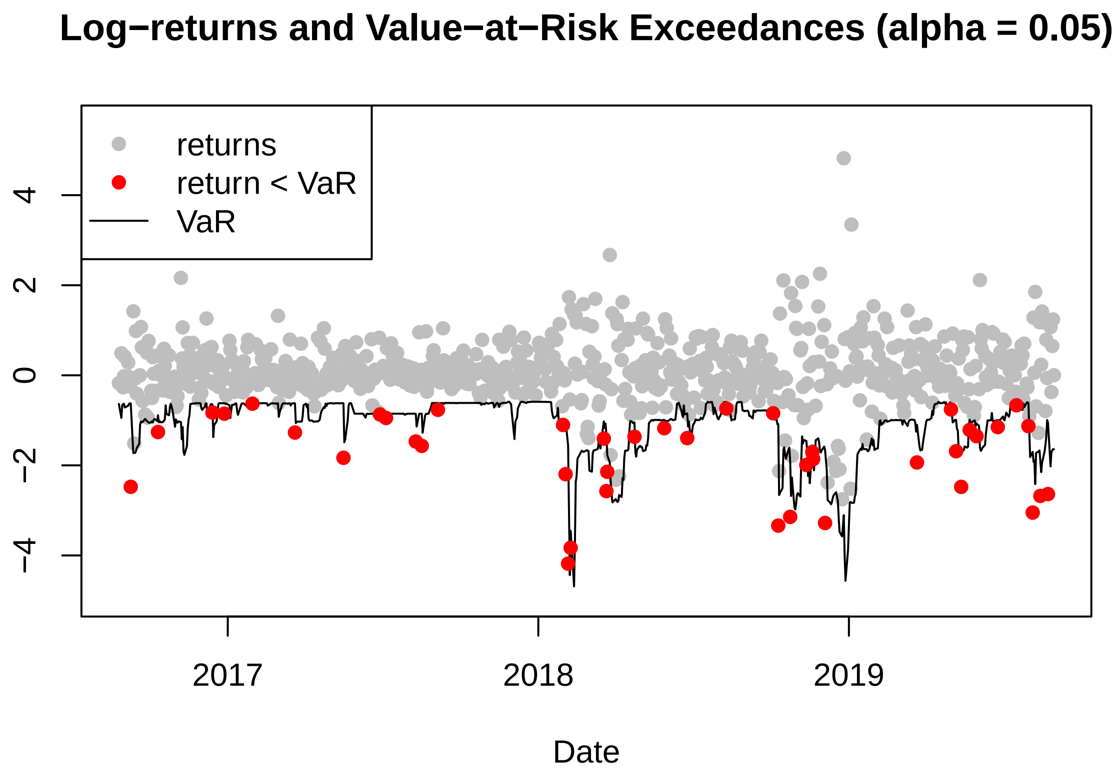
```
## Reject Null        : No
##
## Independance (Christoffersen)
## Null-Hypothesis    : Independance of failures
## LR.ind Statistic   : 0.198
## LR.ind Critical    : 3.841
## LR.ind p-value     : 0.657
## Reject Null        : No
##
## Conditionnal Coverage (Christoffersen)
## Null-Hypothesis    : Correct exceedances and Independance of failures
## LR.cc Statistic    : 0.673
## LR.cc Critical     : 5.991
## LR.cc p-value      : 0.714
## Reject Null        : No
##
## ------------------------------------------------
## alpha              : 0.1%
## Excepted Exceed    : 75.6
## Actual VaR Exceed  : 76
## Actual %           : 0.1%
##
## Unconditionnal Coverage (Kupiec)
## Null-Hypothesis    : Correct exceedances
## LR.uc Statistic    : 0.002
## LR.uc Critical     : 2.706
## LR.uc p-value      : 0.961
## Reject Null        : No
##
## Independance (Christoffersen)
## Null-Hypothesis    : Independance of failures
## LR.ind Statistic   : 1.253
## LR.ind Critical    : 2.706
## LR.ind p-value     : 0.263
## Reject Null        : No
##
## Conditionnal Coverage (Christoffersen)
## Null-Hypothesis    : Correct exceedances and Independance of failures
## LR.cc Statistic    : 1.256
## LR.cc Critical     : 4.605
## LR.cc p-value      : 0.534
## Reject Null        : No
```

```r
# plot
plot(out, plot.type=c("VaR","sigma","dens"))
```

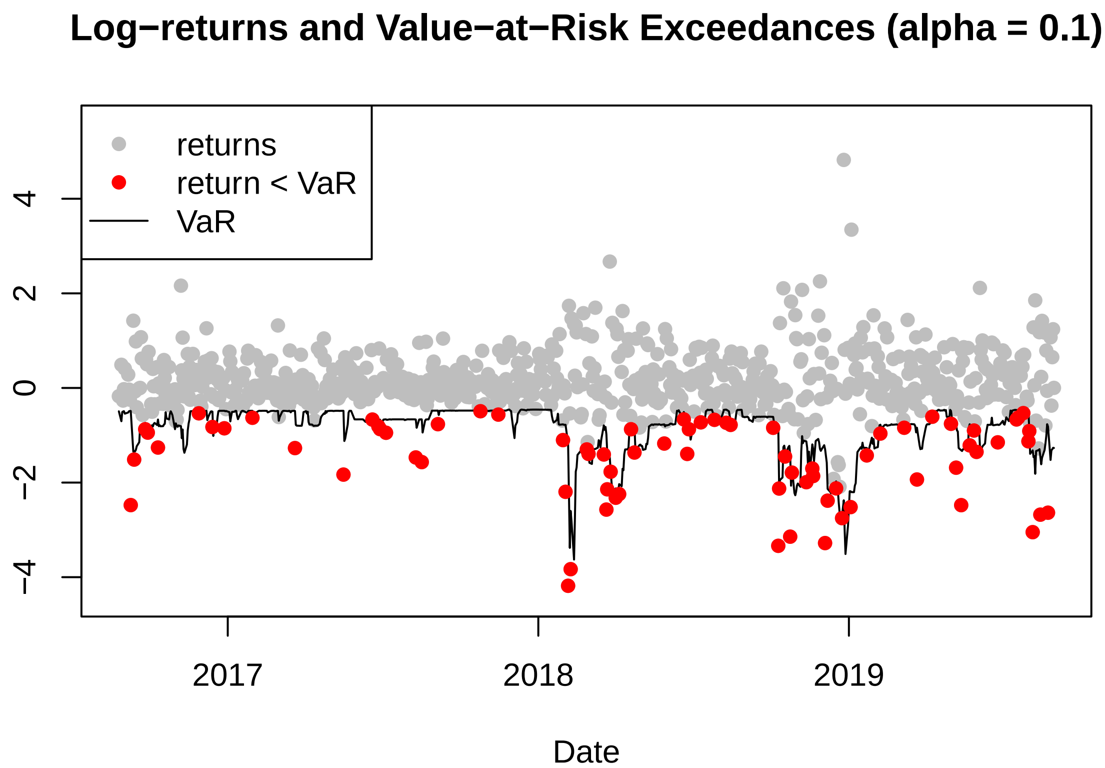

Log−returns square : 1.ahead forecast vs realized values
1.ahead forecast
realized values
0
5
10
15
20
2017
2018
2019
Date

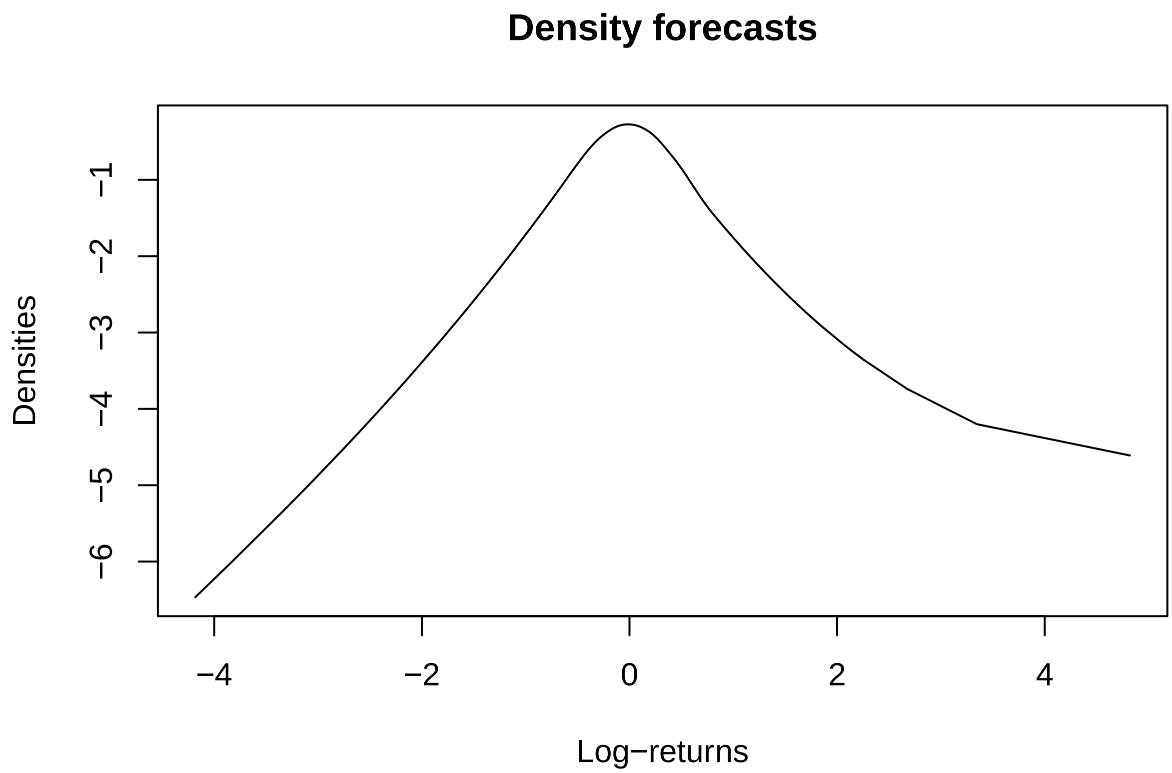

Realized Variances : 1.ahead forecast vs realized values
1.ahead forecast
realized values
0
2
4
6
2017
2018
2019
Date


Log−returns and Value−at−Risk Exceedances (alpha = 0.01)
returns
return < VaR
VaR
6
4
2
0
−2
−4
−6
2017
2018
2019
Date



Log−returns and Value−at−Risk Exceedances (alpha = 0.05)
returns
return < VaR
VaR
4
2
0
−2
−4
2017
2018
2019
Date


Log−returns and Value−at−Risk Exceedances (alpha = 0.1)
returns
return < VaR
VaR
4
2
0
−2
−4
2017
2018
2019
Date



4 Conclusion

This paper provides technical details on the package **MDSV**. It shows with simple and practical examples how to use the package through each of its functions.

References

Augustyniak, M., Bauwens, L., and Dufays, A. (2019). A new approach to volatility modeling: the factorial hidden markov volatility model. *Journal of Business & Economic Statistics*, 37(4):696–709.

Calvet, L. E. and Fisher, A. J. (2004). How to forecast long-run volatility: Regime switching and the estimation of multifractal processes. *Journal of Financial Econometrics*, 2(1):49–83.

Cordis, A. S. and Kirby, C. (2014). Discrete stochastic autoregressive volatility. *Journal of Banking & Finance*, 43:160–178.

Fleming, J. and Kirby, C. (2013). Component-driven regime-switching volatility. *Journal of Financial Econometrics*, 11(2):263–301.

Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press, New Jersey.

Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1):246–256.