Getting started manual for R package **SV** - Quasi-likelihood and indirect inference in non-Gaussian stochastic volatility models

Øivind Skare
Arvid Raknerud

April 9, 2011

1 Introduction

SV is a user friendly R package for doing inference for stochastic volatility models (based on non-Gaussian Ornstein-Uhlenbeck processes) for univariate and bivariate exchange rate data. For univariate exchange rate data both quasi-likelihood estimation and indirect inference are implemented, while for the bivariate data a quasi-likelihood method is implemented. The two papers Raknerud & Skare (2011) and Raknerud & Skare (2010) give an description of these methods. Indirect inference makes use of a large number of quasi-likelihood evaluations and may be cpu demanding. In the Raknerud & Skare (2011) paper, median computing time was somewhat less than one CPU hour for the most complex model considered, which used a simulated time series of 50,000 observations.

The R package interfaces efficient C++ code to ensure a fast implementation of quasi-likelihood and indirect inference.

This manual briefly describes the different functions of SV and gives examples of their usage. R commands are written in verbatim and comments are indicated by `##`, for example:

```
library(SV) ## This command loads the package SV
```

2 Installation

The latest version of R can be downloaded from the R project website (`http://www.r-project.org`).

The R package SV and its documentation are available at the Comprehensive R Archive Network (CRAN) (`http://cran.r-project.org/mirrors.html`). Example data files may be downloaded from the home page `http://folk.uio.no/skare/SV`. For the moment SV is only available for linux. It requires the following freeware to be installed:

- Armadillo (`http://arma.sourceforge.net/download.html`). This is a C++ linear algebra library.

SV is then installed by first downloading the tar file SV_<version>.tar.gz (for the moment SV_1.3.5.tar.gz) from the CRAN page and save it in a suitable directory. You do not need to unpack it. Start R and use, for instance, install.packages(repos = NULL, pkgs = "SV.tar.gz")

To start using SV, use the R command library(SV). SV is then loaded and ready for use. Every time you start a new R session you must load SV with the R command library(SV). (However, you do not need to install from the zip/tar file more than once.)

3 Quasi-likelihood estimation for univariate data

A quasi likelihood is formed by an approximate linear state space representation of the OU-based model. The example data file 'example1.dat' may be downloaded from the home page. The quasi-likelihood estimates are obtained by using the `QL` function. The user must specify data file and start values for the μ (`mu`), ξ (`xi`), λ (`lambda`) and ω (`omega`) parameters. Other optional arguments are documented in the R documentation.

```
help(QL) # R documentation of the QL function
obj <- QL("example1.dat", mu=0.0, xi=0.1, lambda=0.05, omega=0.1)
summary(obj) ## estimates and information about the optimisation
##Summary of quasi-likelihood object
##Number of function evaluations:  19
##Number of gradient evaluations:  19
##Cpu time used:  0.34
##Number of iterations:  9
##Coefficients:
##         Estimate Std. Error   Lower  Upper
##mu        -0.0039     0.0229 -0.0488 0.0410
##xi         0.1070     0.0138  0.0834 0.1372
##lambda_1   0.0822     0.4068  0.0011 1.5522
##omega_1    0.1035     0.0544  0.0428 0.2503
```

4 Indirect inference for univariate data

We do here indirect inference on the example data by means of the function `IndirectInference`. In addition to the data file name, the user must supply start values for the μ (mu), ξ (xi), λ (lambda) and ω (omega) parameters. The indirect inference estimator is obtained by minimizing, in a weighted mean squared error sense, the score vector of the quasi-likelihood function for the simulated data, when this score vector is evaluated at the quasi-likelihood estimator obtained from the real data. (See paper Raknerud & Skare (2011)). Robust standard deviations and confidence intervals are constructed using a sandwich correction.

In this example, the length of the simulated log-return series are twice the number of observations (2000). The accuracy increases with the length of the simulated series. The indirect inference may be computational demanding, increasing linearly with length of simulated time series. This example takes approximately 40 seconds (Intel(R) Xeon(R) E5420 2.50GHz processor).

```
help(IndirectInference) # R documentation of the QL function
resind <- IndirectInference("example1.dat", nTimes=4000, mu=0, xi=0.1,
                            lambda=0.05, omega=0.1)

summary(resind) # estimates and information about the optimisation
```

```
##Summary of indirect inference object
##Number of outer function evaluations:  1000
##Number of function evaluations:  1105
##Number of gradient evaluations:  1105
##Cpu time used:  40.238
##Number of iterations:  23
##Convergence:  OK
##Function value:  0.0009050059
##          Estimate Std. Error   Lower  Upper
##mu          0.0114     0.0285 -0.0443 0.0671
##xi          0.1117     0.0180  0.0819 0.1523
##lambda_1    0.0652     0.1059  0.0091 0.3991
##omega_1     0.1520     0.0489  0.0843 0.2741
```

5 Quasi-likelihood estimation for multivariate data

A quasi likelihood is formed by an approximate linear state space representation of the multivariate OU-based model. The example data file 'exampleMulti.dat' may be downloaded from the home page. The quasi-likelihood estimates are obtained by using the `QLmulti` function. The user must specify data file and start values for the μ (`mu`), ξ (`xi`), λ (`lambda`), ω (`omega`) and ϕ_{21} (`phi21`) parameters. Other optional arguments are documented in the R documentation.

```
help(QLmulti) # R documentation of the QLmulti function

## Simulated data set, Start values = true parameter values
obj.qlmulti <- QLmulti("exampleMulti.dat", mu=rep(0, 2), xi=c(0.4,0.4,0.1),
                       lambda=c(0.005, 0.005, 0.28), omega=c(0.1, 0.1, 0.06),
                       phi21=1, addPenalty=FALSE)

summary(obj.qlmulti) ## estimates and information about the optimisation
##Summary of quasi-likelihood object
##Number of function evaluations:  50
##Number of gradient evaluations:  50
##Cpu time used:  10.6
##Number of iterations:  35
##Coefficients:
##            Estimate Std. Error   Lower  Upper
##mu_1         -0.0146     0.0200 -0.0537 0.0245
##mu_2          0.0043     0.0198 -0.0346 0.0432
##xi_1          0.4323     0.0162  0.4015 0.4655
##xi_2          0.5348     0.0258  0.4863 0.5882
##xi_3          0.0044     0.2862  0.0001 0.3083
##lambda_1_1    0.0032     0.0125  0.0003 0.0351
##lambda_2_1    0.0051     0.0088  0.0008 0.0306
##lambda_3_1    0.3041     0.2340  0.0691 0.9469
##omega_1_1     0.1072     0.3015  0.0127 0.9074
```

```
##omega_2_1     0.2177      0.0508  0.1403 0.3378
##omega_3_1     0.0499      0.3525  0.0031 0.8028
##phi21         0.8342      0.7539 -0.6403 2.3088
```

The bivariate data in 'exampleMulti.dat' are shown in figure 1 together with the underlying latent volatility processes.

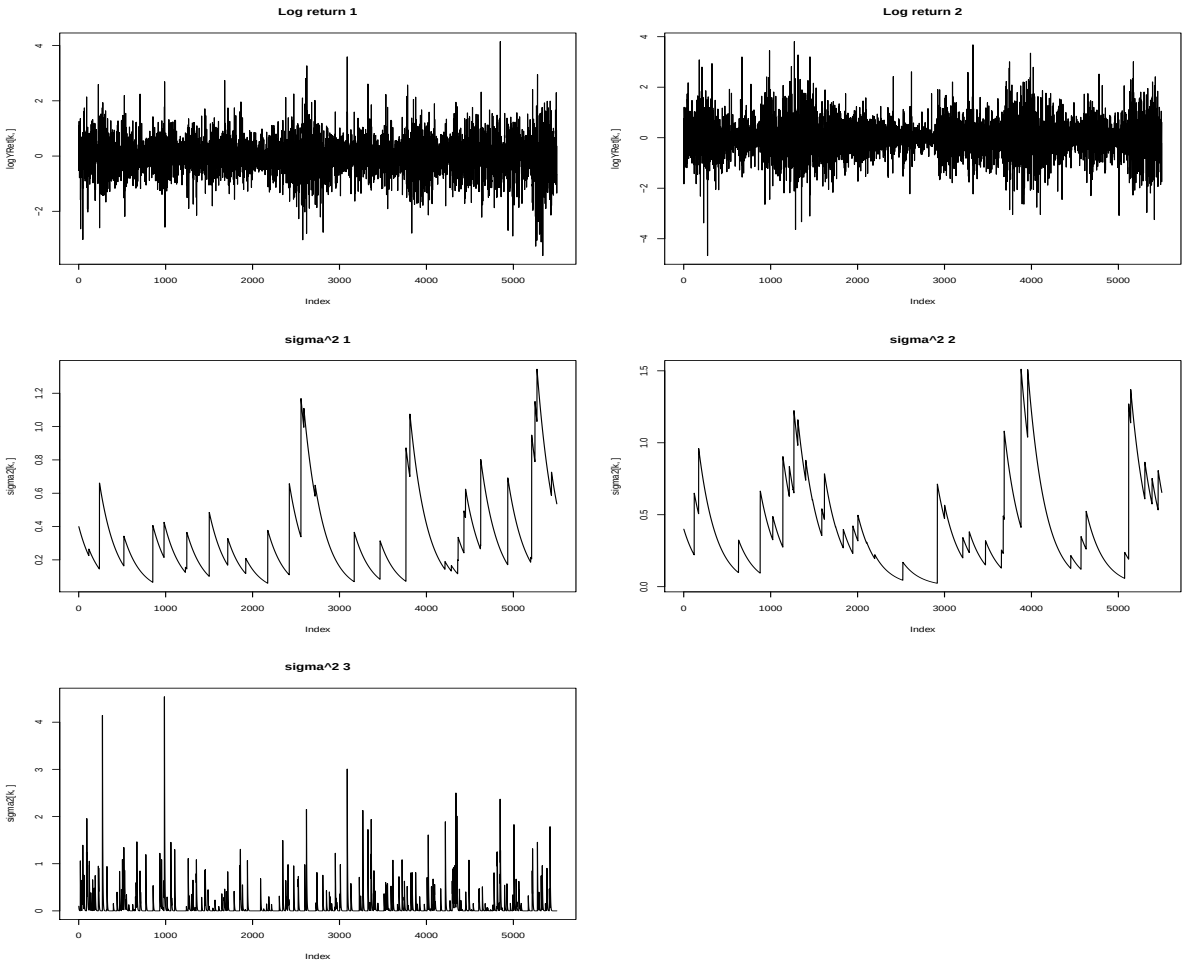

Figure 1: The simulated bivariate example data (exampleMulti.dat) and the three corresponding latent volatility processes.

6 Simulating and plotting stochastic volatility data

The functions `SimulateVolatility` and `SimulateVolatilityMulti` simulates and plots a volatility series σ_n^2 and log-return data for the univariate and multivariate model, respectively.

```
## R documentation of the SimulateVolatility function for univariate data
help(SimulateVolatility)
```

```
## R documentation of the SimulateVolatility function for bivariate data
help(SimulateVolatilityMulti)

## Simulates and plots the sigma^2 and log-return data:
obj <- SimulateVolatility(1, 5000, mu=0.015, xi=0.5, lambda=0.1, omega=0.1)
## obj contains the sigma^2 and log-return data  (obj$sigma2, obj$logYRet)

par(mfrow=c(3,2))
objmulti <- SimulateVolatilityMulti(1, 5500, mu=rep(0, 2), xi=c(0.4,0.4,0.1),
                                    lambda=c(0.005, 0.005, 0.28),
                                     omega=c(0.1, 0.1, 0.06), phi21=1)
## obj contains the sigma^2 and log-return data.
##    obj$sigma2[k,]: k'th sigma^2 process (k=1,2,3)
##    obj$logYRet[k,]: k'th log-return data (k=1,2)
```

References

Raknerud, A. & Skare, Ø. (2010), 'Multivariate stochastic volatility models based on non-Gaussian Ornstein-Uhlenbeck processes', *Discussion Papers No. 614, March, Statistics Norway, Research Department* .

Raknerud, A. & Skare, Ø. (2011), 'Indirect inference methods for stochastic volatility models based on non-Gaussian Ornstein-Uhlenbeck processes', *Computational Statistics & Data Analysis* . Accepted for publication.