Markov-Switching GARCH Models in R: The MSGARCH Package

David Ardia
University of Neuchâtel
Laval University

Keven Bluteau
University of Neuchâtel

Kris Boudt
Vrije Universiteit Brussel
Vrije Universiteit Amsterdam

Brian G. Peterson
Black Rock

VERY PREMIMINARY
PLEASE DO NOT QUOTE

Abstract

This paper introduces the R package **MSGARCH** for calibration and simulation of Markov-switching GARCH models. We provide an empirical illustration to real financial data.

Keywords: GARCH, MSGARCH, Markov-switching, conditional volatility, risk management, R sofware.

1. Introduction

A seminal contribution in financial econometrics was the development of the GARCH model by Bollerslev (1986) where a time-varying conditional variance is expressed as a linear function of past squared returns and past conditional variances. The GARCH model is today a widespread tool for dealing with the heteroscedasticity observed in financial time series. However, estimation of GARCH models often leads to a very high persistence or even nonstationarity in the conditional variance (Caporale *et al.* 2003). An explanation proposed in the literature is the presence of structural breaks in the conditional variance dynamics (Lamoureux and Lastrapes 1990; Mikosch and Stărică 2004; Hillebrand 2005). These structural breaks typically occur during periods of financial turmoil. Estimating a regular GARCH model on data displaying one or more structural breaks yields a substantially misspecified model, which may imply very poor risk predictions. A way to cope with this problem is provided by Markov-switching GARCH (MSGARCH) whose parameters vary over time according to some regimes. These models can quickly adapt to variations in the unconditional volatility level, which improves risk predictions (see Ardia 2008).

The R package **MSGARCH** aims to provide a comprehensive set of methods for the estimation, the simulation and the forecasting of MSGARCH models. Also, methods for risk management such as Value-at-Risk and Expected-Shortfall calculations are available. In this vignette, we describe the models and the functions/methods available in the package.

2. Model specification

We provide in this section the steps to create simple or complex specifications using the function `create.spec`. The R package **MSGARCH** supports single-regime models as they are the building blocks for multiple-regime models. The simplest specification we can build is a GARCH model with a symmetric Normal conditional distribution:

```
R> spec = create.spec(model = "sGARCH", distribution = "norm", do.skew = FALSE)
```

The simplest MSGARCH process we can create is a two regime GARCH process with each regime having a symmetric Normal conditional distributions:

```
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("norm", "norm"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = FALSE, do.shape.ind = FALSE)
```

The argument `model` takes one or more single-regime specifications describing each regime conditional variance process while the argument `distribution` contains each regime respective conditional distribution. Valid models (see Section 2.1) are `"sGARCH"`, `"eGARCH"`, `"gjrGARCH"`, `"tGARCH"`, and `"GAS"`. All single-regime conditional variance processes are one-lag processes (e.g., GARCH(1,1)). One-lag conditional variance processes has proved to be sufficient in financial econometrics and it reduces models complexity. Valid conditional distributions (see Section 2.3) are `"norm"`, `"std"`, and `"ged"`. Each conditional distribution have a skewed version (see Section 2.3.4) which can be selected by setting the argument `do.skew = TRUE`. The argument `do.mix` allow the user to create a Mixture of GARCH process (see Section 2.2.2) instead of a MSGARCH process (see Section 2.2.1). Finally, the the `do.shape.ind` argument allows for regime-independent shape parameters (see Section 2.2.3).

The user can technically create an infinite amount of different specification by combining and adding single-regime models. See for example this complex three state MSGARCH process:

```
R> spec = create.spec(model = c("sGARCH", "tGARCH", "eGARCH"),
                            distribution = c("norm", "std", "ged"),
                            do.skew = c(TRUE, FALSE, TRUE),
                            do.mix = FALSE, do.shape.ind = FALSE)
```

However, care should be taken when adding complexity to the specification since reliable optimization can become very difficult (see Section 3 for more details).

The output of the function `create.spec` is a list of class `MSGARCH_SPEC` which contains functions and variables. The relevant information are summarize in the `print` or `summary` function.

```
R> spec = create.spec()
R> print(spec)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: sGARCH_normal_sym sGARCH_normal_sym"
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in each distribution: 0 0"
[1] "Default parameters:"
     alpha0 alpha1 beta alpha0 alpha1 beta   P   P
[1,]    0.1    0.1  0.8    0.1    0.1  0.8 0.5 0.5
```

2.1. Single-regime specifications

We present in this section the various single-regime specifications available in the R package **MSGARCH**.

GARCH model

The GARCH model by Bollerslev (1986) can be written as:

$$y_t = \eta h_t^{1/2} \tag{1}$$

$$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \beta h_{t-1} \,, \tag{2}$$

where $\eta \sim i.i.d. \, \mathcal{D}(0, 1, \lambda)$ with $\mathcal{D}$ a distribution with zero mean, unit variance, and shape parameters λ. To ensure positivity, we require $\alpha_0 > 0$, $\alpha_1 \geq 0$, $\beta \geq 0$. Covariance-stationarity is obtained by adding the condition $\alpha_1 + \beta < 1$. To create a single-regime GARCH specification we use `model = "sGARCH"` in the function `create.spec`.

EGARCH model

The Exponential GARCH (EGARCH) of Nelson (1991) can be written as:

$$\ln(h_t) \equiv \alpha_0 + \alpha_1 \big(|y_{t-1}| - \mathsf{E}[|y_{t-1}|]\big) + \alpha_2 y_{t-1} + \beta \ln(h_{t-1}) \,, \tag{3}$$

where the natural logarithm of conditional variance $\ln(h_t)$ is modeled instead of h_t. This model takes into consideration the leverage effect where past negative returns have a larger influence on the conditional volatility than past positive returns of the same magnitude (Black 1976; Christie 1982). The persistence of the models is captured by the coefficient β. The creation of a single-regime EGARCH specification is done by using `model = "eGARCH"` in the function `create.spec`.

GJR model

Glosten *et al.* (1993) have developed the GJR model that also captures the asymmetry in the conditional volatility process:

$$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \alpha_2 y_{t-1}^2 \mathbb{I}_{y_{t-1}<0} + \beta h_{t-1} \,, \tag{4}$$

where $\mathbb{I}_{y_t \geq 0} \equiv 0$ if $y_t \geq 0$ and $\mathbb{I}_{y_t<0} \equiv 1$ otherwise. The parameter α_2 controls the degree of asymmetry. To ensure positivity, we usually set $\alpha_0 > 0$, $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, $\beta \geq 0$ (sufficient condition). To ensure covariance-stationarity we make sure that $\alpha_1 + \alpha_2 \mathsf{E}[\eta^2 \mathbb{I}_{\eta<0}] + \beta < 1$.

The single-regime GJR specification is created by using `model = "gjrGARCH"` in the function `create.spec`:

TGARCH model

Zakoian (1994) introduces the TGARCH which has the conditional volatility as dependent variable instead of the conditional variance:

$$h_t^{1/2} \equiv \alpha_0 + \alpha_1 y_{t-1} \mathbb{I}_{y_{t-1} \geq 0} + \alpha_2 y_{t-1} \mathbb{I}_{y_{t-1} < 0} + \beta h_{t-1}^{1/2} \,. \tag{5}$$

For positivity we set $\alpha_0 > 0$, $\alpha_1 \geq 0$, $\alpha_2 \geq 0$ and $\beta \geq 0$. To ensure covariance-stationarity, we make sure that $\alpha_1^2 + \beta^2 - 2\beta(\alpha_1 + \alpha_2)\mathsf{E}[\eta \mathbb{I}_{\eta<0}] - (\alpha_1^2 - \alpha_2^2)\mathsf{E}[\eta^2 \mathbb{I}_{\eta<0}] < 1$ (see Francq and Zakoian 2011, Section 10.2). The single-regime TGARCH specification is created by using `model = "tGARCH"` in function `create.spec`:

GAS model

Generalized Autoregressive Score models were proposed in their full generality in Creal *et al.* (2013). It provides a general framework for modeling time variation in parametric models. The GAS model can be written as:

$$h_t \equiv \alpha_0 + \alpha_1 s_{t-1} + \beta h_{t-1} , \quad s_{t-1} \equiv S_{t-1} \nabla_{t-1} , \quad \nabla_{t-1} \equiv \frac{\partial \ln f(y_{t-1} | h_{t-1}, \lambda)}{\partial h_{t-1}} \,, \tag{6}$$

where $f(y_{t-1}|h_{t-1}, \lambda)$ is the likelihood of y_{t-1} given h_{t-1} and the distribution's shape parameters λ, s_{t-1} is the score function, and S_{t-1} is a scaling function for the score of the observation log-density. The scaling function in this case is defined as:

$$S_{t-1} \equiv \mathsf{E}[\nabla_{t-1} \nabla'_{t-1}]^{-1} \,. \tag{7}$$

The single-regime GAS model is created by using `model = "GAS"` in the function `create.spec`.

2.2. Multiple-regime specifications

We present in this section the two multiple-regime specifications available in the R package **MSGARCH**.

Markov-switching GARCH

Suppose Δ_t is a Markov chain with a finite state space $S \equiv \{1, 2, ..., K\}$ with an irreducible and primitive $K \times K$ transition matrix $\mathbf{P}$ defined as:

$$\mathbf{P} \equiv \begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{K,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,K} & p_{2,K} & \cdots & p_{K,K} \end{bmatrix} , \tag{8}$$

where $0 \leq p_{i,j} \leq 1$ is the probability of switching from state $\Delta_{t-1} = i$ to state $\Delta_t = j$ and $\sum_{j=1}^{K} p_{i,j} = 1$ $(i = 1, \ldots, K)$.

Let the returns of a financial asset at time t be expressed as:

$$y_t = \eta_{\Delta_t} h_{\Delta_t,t}^{1/2} \,, \tag{9}$$

where $\eta_{\Delta_t} \sim i.i.d.\mathcal{D}_{\Delta_t}(0, 1, \lambda_{\Delta_t})$, with $\mathcal{D}_{\Delta_t}$ a distribution with zero mean, unit variance, and shape parameters λ_{Δ_t}, and $h_{\Delta_t,t}^{1/2}$ the conditional variance in state Δ_t at time t. For a the single-regime specification in state k, we define θ_k as the parameters of the conditional variance process, λ_k as the shape parameters of the conditional distribution $\mathcal{D}_k$, and the $(T \times 1)$ vector $\boldsymbol{h}_k \equiv (h_{k,1}, h_{k,2}, \ldots, h_{k,T})'$ as the resulting conditional variance vector from this specification. The MSGARCH specification is constructed following the approach by Haas *et al.* (2004a), which consists of many distinct single-regime specifications evolving in parallel. We define $\Theta \equiv [\theta_1, \theta_2, ..., \theta_K]$, $\boldsymbol{D} \equiv [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K]$, $\Lambda \equiv [\lambda_1, \lambda_2, ..., \lambda_K]$ and $\mathbf{H} \equiv [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K]$.

We create a two-state MSGARCH model from two single-regime GARCH processes each following a Normal distribution. We fit the model to the `sp500` dataset which consists of the S&P 500 index closing value log-returns ranging from 1998-01-01 to 2015-12-31.

```
R> data("sp500")
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("norm", "norm"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = FALSE, do.shape.ind = FALSE)
R> ctr.mle = list(do.init = TRUE, NP = 10*length(spec$theta0), itermax = 500,
                  do.enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> summary(fit.mle)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: sGARCH_normal_sym sGARCH_normal_sym"
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in each distribution: 0 0"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,]      0.1      0.1    0.8      0.1      0.1
     beta_2   P   P
[1,]    0.8 0.5 0.5
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
     alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,] 0.001601  0.02711 0.8913  0.03795   0.1177
     beta_2         P      P
[1,] 0.8778 3.212e-07 0.3704
[1] "Transition matrix:"
                t = 1  t = 2
t + 1 = 1 3.212e-07 0.3704
t + 1 = 2 1.000e+00 0.6296
[1] "Stable probabilities:"
```

```
        Stable probabilities
State 1               0.2703
State 2               0.7297
[1] "Unconditional volatility:"
     State 1 State 2
[1,]    0.14   2.892
[1] "Log-kernel: -6506.63976020117"
[1] "AIC: 13235.894384861"
[1] "BIC: 13287.2404364094"
```

Model is fitted by Maximum likelihood with the function `fit.mle` (see Section 3.0.1). The resulting parameters are collected in the vector `theta` where each parameter are labeled according to the model and their state. The function `transmat` is an helper function that builds the transition matrix from the fitted parameters for better readability.

Mixture of GARCH

Haas *et al.* (2004b) propose a general class of Mixture of GARCH models. They specify a Mixture of Normal distributions where the variance process of each Normal component is a GARCH process. They name this new class the MNGARCH models. A special case of this specification named the Full and Diagonal MNGARCH is encountered when all the covariance between each component is constrained to be zero. This special case has a direct relationship with the MSGARCH model. Indeed, we can constrain the transition matrix $\mathbf{P}$ of the MSGARCH model to make the probability $p_{i,j}$ of switching from any state $\Delta_{t-1} = i$ to state $\Delta_t = j$ the same. That is, $P(\Delta_t = j | \Delta_{t-1} = i) \equiv p_j$ $(i = 1, \ldots, K)$. The transition matrix reduces then to a probability vector $\mathbf{P} \equiv [p_1, p_2, ..., p_K]$. This constraint converts the Markov-switching behavior to a Mixture behavior since the probabilities do not depend on the current state. For demonstration, lets repeat the experiment done previously, but with the argument `do.mix = TRUE`.

```
R> data("sp500")
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("norm", "norm"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = TRUE, do.shape.ind = FALSE)

R> ctr.mle = list(do.init = TRUE, NP = 10*length(spec$theta0), itermax = 500,
                  do.enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> summary(fit.mle)
```

```
[1] "Specification Type: Mixture"
[1] "Specification Name: sGARCH_normal_sym sGARCH_normal_sym"
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in each distribution: 0 0"
[1] "Default parameters:"
```

```
     alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,]      0.1      0.1    0.8      0.1      0.1
     beta_2   P
[1,]    0.8 0.5
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
      alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,] 0.0003265  0.01765 0.9099  0.02865   0.1067
     beta_2     P
[1,] 0.8886 0.199
[1] "Stable probabilities:"
        Stable probabilities
State 1                0.199
State 2                0.801
[1] "Unconditional volatility:"
     State 1 State 2
[1,] 0.06712   2.472
[1] "Log-kernel: -6518.18297178244"
[1] "AIC: 13188.2083263841"
[1] "BIC: 13233.136121489"
```

We can observe that we have less parameters label as `P` since a Mixture of GARCH will always have less parameters than a Markov-Switching GARCH process. The `transmat` function, for a Mixture of GARCH, will output a probability vector and not a probability matrix.

Regime-independent shape parameters

Sometimes it is useful to have a regime-switching behavior only in the conditional variance and keep the same conditional distribution across regimes. We call this regime-independent shape parameters since all distributions $\mathcal{D}_k$ in $\boldsymbol{D}$ and λ_k in Λ are restricted to be the same (i.e., they only differ via the conditional variance process of each regime). This can be done by setting the parameter `do.shape.ind = TRUE`. We illustrate this with a two-state MSGARCH model with two single-regime GARCH processes following the same Student-t distribution.

```
R> data("sp500")
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("std", "std"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = FALSE, do.shape.ind = TRUE)
R> ctr.mle = list(do.init = TRUE, NP = 10*length(spec$theta0), itermax = 500,
                  do.enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> summary(fit.mle)
```

```
[1] "Specification Type: Markov-Switching with Regime-Independent distribution"
[1] "Specification Name: sGARCH_student_sym sGARCH_student_sym"
```

```
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in distribution: 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,]      0.1      0.1    0.8      0.1      0.1
     beta_2 nu_1   P   P
[1,]    0.8   10 0.5 0.5
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
     alpha0_1 alpha1_1 beta_1 alpha0_2 alpha1_2
[1,] 0.000298  0.02191   0.88  0.02331   0.1016
     beta_2  nu_1         P      P
[1,] 0.8938 15.73 4.918e-06 0.1838
[1] "Transition matrix:"
             t = 1  t = 2
t + 1 = 1 4.918e-06 0.1838
t + 1 = 2 1.000e+00 0.8162
[1] "Stable probabilities:"
        Stable probabilities
State 1               0.1553
State 2               0.8447
[1] "Unconditional volatility:"
     State 1 State 2
[1,] 0.05512   2.232
[1] "Log-kernel: -6505.99800787707"
[1] "AIC: 13121.7705309498"
[1] "BIC: 13179.5348389418"
```

As we can see, the output only contains one parameter `nu` with no regime indication instead of two parameters `nu_1` and `nu_2`.

2.3. Distributions

We present here the conditional distributions and their functionalities available in the R package **MSGARCH**. There are two functions directly related to the conditional distribution: The probability density function (PDF) and the cumulative density function (CDF):

```
cdf(object, x, theta, y, is.log, do.its)
pdf(object, x, theta, y, is.log, do.its)
```

First, the `object` argument accepts a specification created with `create.spec` (see Section 2) or a fitted object (see Section 3). When a specification is passed to the method, `theta` and `y` must be provided. This is not the case when a fitted object is passed as the fitted object already has relevant `theta` (which is found during optimization) and `y` (which is used in the optimization). The `cdf` and `pdf` functions can be used in two different ways via the `do.its` argument. When `do.its = TRUE`, we do an in-sample evaluation of `y`, that is: $F(\mathbf{y}|\mathbf{H},\boldsymbol{\psi},\Lambda)$ or $f(\mathbf{y}|\mathbf{H},\boldsymbol{\psi},\Lambda)$ where $F(\cdot)$ is for the CDF, $f(\cdot)$ is for the PDF, and $\boldsymbol{\psi} \equiv [\psi_1, \psi_2, ..., \psi_T]$ where

ψ_t are the vector of conditional probabilities to be in each state at time t (see Section 3). When `do.its = FALSE`, we evaluate the points `x` as $T+1$ realizations, that is: $F(\mathbf{x}|H_{T+1}, \psi_{T+1}, \Lambda)$ or $f(\mathbf{x}|H_{T+1}, \psi_{T+1}, \Lambda)$ where $H_{T+1} \equiv [h_{T+1,1}, h_{T+1,2}, ..., h_{T+1,K}]$.

The Normal distribution

The PDF of the standardized Normal distribution can be written as:

$$f_{\mathcal{N}(0,1)}(z) \equiv \frac{1}{\sqrt{2\pi}} \; \mathrm{e}^{-\frac{1}{2}z^2} \,, \tag{10}$$

where $z \equiv \frac{x-\mu}{\sigma}$. The Normal distribution is completely described by its first two moments: the mean and the variance. The distribution is symmetric. The CDF is calculated with the function `pnorm` from the `stats` namespace. To create any specification with a symmetric Normal distribution we use `distribution = "norm"` in the function `create.spec`.

The Student-t distribution

The PDF of the standardized Student-t distribution can be written as:

$$f_{\mathcal{S}(0,1,\nu)}(z) \equiv \sqrt{\frac{\nu}{\nu-2}} \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{z^2}{\nu}\right)^{-\frac{\nu+1}{2}} \,, \tag{11}$$

where Γ is the Gamma function (we use the `gamma` function from the `base` namespace) and $\nu > 2$ is the shape parameter. It is completely described by the shape parameter ν. The kurtosis of a Student-t distribution is higher for lower ν. For $\nu = \infty$, the Student-t distribution is equivalent to the Normal distribution. The distribution is symmetric. The CDF is calculated with the function `pt` from the `stats` namespace. To create any specification with a symmetric Student-t distribution we use `distribution = "std"` in the function `create.spec`.

The GED distribution

The PDF of the standardized GED distribution can be written as:

$$f_{\mathcal{GED}(0,1,\nu)}(z) \equiv \frac{\nu e^{-\frac{1}{2}|z/\lambda|^{\nu}}}{\lambda 2^{(1+1/\nu)}\Gamma(1/\nu)} \,, \tag{12}$$

where $\lambda \equiv [2^{-2/\nu}\Gamma(1/\nu)/\Gamma(3/\nu)]^{1/2}$. As in the Student-$t$ distribution, the GED distribution is described completely by the shape parameter ν. As ν decreases the density gets flatter. Special cases are the Normal distribution when $\nu = 2$ and the Laplace distribution when $\nu = 1$. The distribution is symmetric. The CDF of the standardized GED distribution can be written as:

$$F_{\mathcal{GED}(0,1,\nu)}(z) \equiv \begin{cases} \frac{1}{2} - \frac{1}{2} F_{\mathcal{GAM}(1,1/\nu)}\left(\frac{1}{2}\left(\frac{|z|}{\lambda}\right)^{\nu}\right) & \text{if} \quad z \leq 0 \\ \frac{1}{2} + \frac{1}{2} F_{\mathcal{GAM}(1,1/\nu)}\left(\frac{1}{2}\left(\frac{z}{\lambda}\right)^{\nu}\right) & \text{if} \quad z \geq 0 \,, \end{cases} \tag{13}$$

where $F_{\mathcal{GAM}(1,1/\nu)}(\cdot)$ is the Gamma distribution CDF with parameter $\beta = 1$ and $\alpha = 1/\nu$. The Gamma distribution CDF is calculated with the function `pgamma` from the `stats` namespace. To create any specification with a symmetric GED distribution we use `distribution = "ged"` in the function `create.spec`.

Skewed distributions

Fernández and Steel (1998) provide a simple way to include skewness into a unimodal standardized distribution. Trottier and Ardia (2016) derive the moments of the standardized Fernandez-Steel skewed distributiosn which are needed in the estimation of the GJR, EGARCH, and TGARCH models. Following Trottier and Ardia (2016), the standardized Fernandez-Steel skewed PDF can be written as:

$$f_{\mathcal{D}skew(\xi,\lambda)} \equiv \frac{2\sigma_\xi^2}{\xi+\xi^{-1}} f_{\mathcal{D}sym(\lambda)}(z_\xi), \quad z_\xi \equiv \begin{cases} \xi^{-1}(\sigma_\xi z + \mu_\xi) & \text{if} \quad z \geq -\mu_\xi/\sigma_\xi \\ \xi(\sigma_\xi z + \mu_\xi) & \text{if} \quad z < -\mu_\xi/\sigma_\xi \,, \end{cases} \tag{14}$$

where $0 < \xi < \infty$ is the parameter describing the degree of asymmetry and $f_{\mathcal{D}sym(\boldsymbol{\lambda})}$ is any symmetric uni-modal PDF with zero mean, unit variance, and shape parameters λ. To create any specification with skewed distribution we use the argument `do.skew = TRUE` in the function `create.spec`.

3. Estimation

Estimation is done by maximizing the negative log-likelihood which corresponds to the negative value of the function `kernel`. The kernel is a combination of the prior and the likelihood function. The kernel is equal to $\text{prior}(\Theta) + \text{prior}(\Lambda) + \text{prior}(\mathbf{P}) + L(\mathbf{y}|\Theta, \Lambda, \mathbf{P})$ where L is the likelihood of $\mathbf{y}$ given the parameter Θ, Λ, and $\mathbf{P}$. The prior is different for each conditional variance process (see Section 2.1), conditional distribution (see Section 2.3), and type of model (see Section 2.2). It ensures that the Θ makes the conditional variance processes stationary and positive, that Λ respect the parameters bounds of all the conditional distributions, and that $\mathbf{P}$ respects that the sums of the probabilities in the case of a multiple-regime models are all equal to one. If any of these conditions is not respected, the prior returns `-1e10`.

The log-likelihood of a single-regime model is:

$$LLH(\theta, \lambda \,|\, \mathbf{y}) \equiv \sum_{t=1}^{T} \ln\left(f_t(y_t \,|\, h_t, \theta, \lambda)\right) \,, \tag{15}$$

where f_t is the conditional density at time t. The log-likelihood for a multiple-regime process is more complex since it has to be calculated with a recursive calculation (Hamilton 1989):

$$LLH(\Theta, \Lambda, \mathbf{P}|\mathbf{y}) \equiv \sum_{t=1}^{T} \ln\left(\boldsymbol{\psi}_t' \mathbf{f}_t(y_t|H_t, \Theta, \Lambda)\right) \,, \tag{16}$$

where the vector $\boldsymbol{\psi}_t$ contains the filtered probabilities to be in each states at time t and the vector $\mathbf{f}_t$ contains the conditional density of each regime at time t. The vector $\boldsymbol{\psi}_t$ is easily computed with the Hamilton filter:

$$\begin{aligned} \mathring{\boldsymbol{\psi}}_t &\equiv \frac{\boldsymbol{\psi}_{t-1} \circ \mathbf{f}_t(y_t|H_t, \Theta, \Lambda)}{\boldsymbol{\iota}'(\boldsymbol{\psi}_{t-1} \circ \mathbf{f}_t(y_t|H_t, \Theta, \Lambda))} \\ \boldsymbol{\psi}_t &\equiv \mathbf{P}\mathring{\boldsymbol{\psi}}_t \,. \end{aligned} \tag{17}$$

The R package **MSGARCH** allows Maximum likelihood and Bayesian estimation of MSGARCH models.

Maximum likelihood estimation

Obtaining the Maximum likelihood estimator of a multiple-regime specification can be difficult using a standard optimization scheme. Because of this, we rely by default on a two-step procedure. We first use differential evolution (Price *et al.* 2006) implemented in the R package **DEoptim** (Mullen *et al.* 2011) as a global optimizer and then use the resulting fitted parameters as initialization for a local optimization using a sequential least-squares quadratic programming algorithm (Kraft 1988) implemented in the function `slsqp` of the R package **nloptr** Johnson (2014).

```
R> data("sp500")
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("std", "std"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = FALSE, do.shape.ind = FALSE)
R> ctr.mle = list(do.init = TRUE, NP = 10*length(spec$theta0),
                  itermax = 500, do.enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> summary(fit.mle)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: sGARCH_student_sym sGARCH_student_sym"
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in each distribution: 1 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 nu_1 alpha0_2
[1,]      0.1      0.1    0.8   10      0.1
     alpha1_2 beta_2 nu_2   P   P
[1,]      0.1    0.8   10 0.5 0.5
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
     alpha0_1 alpha1_1 beta_1  nu_1 alpha0_2
[1,] 0.004169  0.05999 0.8968 3.351  0.02612
     alpha1_2 beta_2 nu_2         P      P
[1,]   0.1037 0.8917   25 2.884e-05 0.3958
[1] "Transition matrix:"
             t = 1  t = 2
t + 1 = 1 2.884e-05 0.3958
t + 1 = 2 1.000e+00 0.6042
[1] "Stable probabilities:"
        Stable probabilities
State 1               0.2836
State 2               0.7164
[1] "Unconditional volatility:"
     State 1 State 2
[1,]  0.3106   2.383
```

```
[1] "Log-kernel: -6503.83678207374"
[1] "AIC: 13080.1083894088"
[1] "BIC: 13144.2909538444"
```

As shown in the example above, we allow the user to directly control some of the argument of `DEoptim` in the list `ctr`. The argument `do.init` indicates if there is a pre-optimization with the R package **DEoptim**. For simpler specifications such as single-regime specifications, setting `do.init = FALSE` and skipping the initialization step should provide a good estimation. The argument `NP` sets the number of vector of parameters in the population while `itermax` sets the maximum number of iterations (number of populations generated). Please refers to the **DEoptim** documentation for more details. Finally, `do.enhance.theta0` uses the volatilities of rolling windows of `y` and adjust the default parameters so that the unconditional volatility of each regime is set to different quantiles of the volatilities obtained with rolling windows on `y`.

Bayesian estimation

To perform Bayesian estimation we use the Adaptive Metropolis-Hastings sampler described in Vihola (2012) and available in the R package **adaptMCMC** (Andreas 2012).

```
R> data("sp500")
R> spec = create.spec(model = c("sGARCH", "sGARCH"),
                      distribution = c("norm", "norm"),
                      do.skew = c(FALSE, FALSE),
                      do.mix = FALSE, do.shape.ind = FALSE)
R> set.seed(123)
R> ctr.bay = list(N.burn = 20000, N.mcmc = 10000, N.thin = 10,
                  do.enhance.theta0 = TRUE)
R> fit.bay= MSGARCH::fit.bayes(spec = spec, y = sp500, ctr = ctr.bay)
R> tail(fit.bay$theta, 5)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: sGARCH_student_sym sGARCH_student_sym"
[1] "Number of parameters in each variance model: 3 3"
[1] "Number of parameters in each distribution: 1 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 nu_1 alpha0_2
[1,]      0.1      0.1    0.8   10      0.1
     alpha1_2 beta_2 nu_2   P   P
[1,]      0.1    0.8   10 0.5 0.5
[1] "Bayesian posterior mean:"
alpha0_1 alpha1_1   beta_1     nu_1 alpha0_2
0.01146  0.08429  0.90644  9.86376  0.55834
alpha1_2   beta_2     nu_2        P        P
0.38338  0.49062 10.11514  0.97443  0.67645
[1] "Posterior variance-covariance matrix"
           alpha0_1   alpha1_1     beta_1
```

```
alpha0_1  1.306e-05  2.052e-05 -2.920e-05
alpha1_1  2.052e-05  1.097e-04 -1.100e-04
beta_1   -2.920e-05 -1.100e-04  1.233e-04
nu_1     -5.745e-05 -5.996e-05  6.688e-05
               nu_1   alpha0_2   alpha1_2
alpha0_1 -5.745e-05  7.725e-05  6.534e-05
alpha1_1 -5.996e-05  1.389e-04  1.365e-04
beta_1    6.688e-05 -1.002e-04 -1.218e-04
nu_1      1.716e-02 -3.352e-03 -6.777e-03
            beta_2       nu_2          P
alpha0_1 -9.290e-05 -0.0000229  3.352e-05
alpha1_1 -3.238e-04 -0.0001215  5.504e-05
beta_1    2.584e-04  0.0001008 -6.261e-05
nu_1      1.597e-02 -0.0045914  3.114e-04
                 P
alpha0_1  6.665e-05
alpha1_1  2.033e-04
beta_1   -1.899e-04
nu_1     -1.810e-02
[ reached getOption("max.print") -- omitted 6 rows ]
[1] "Posterior mean transition matrix:"
            t = 1  t = 2
t + 1 = 1 0.97443 0.6764
t + 1 = 2 0.02557 0.3236
[1] "Posterior mean stable probabilities:"
        Stable probabilities
State 1              0.96357
State 2              0.03643
[1] "Posterior mean unconditional volatility:"
     State 1 State 2
[1,]   1.112   2.105
[1] "Acceptance rate: 0.986"
[1] "AIC: 13006.9565090122"
[1] "BIC: 13071.1390734477"
[1] "DIC: 13003.1027841308"
```

The function `fit.bayes` takes up to five controls arguments in `ctr`. The argument `N.mcmc` is the number of draws to keep, `N.burn` is number of discarded draws, and `N.thin` is the thinning factor. `N.burn` and `N.thin` main purpose is to remove auto-correlation in the chain. The argument `N.burn` also serve as pre-optimization which is why `N.burn` is so high in the example. This is due to the fact that the chain begins with the specification default parameters which are not good estimators and it can take many iterations before converging to stable estimators. One alternative is to use a custom starting parameters `theta0` in the `ctr` argument or to set `do.enhance.theta0 = TRUE`. For example, we could set `theta0` as the Maximum likelihood estimator estimated with `fit.mle`. The total length of the chain is: `N.mcmc / N.thin`. The chain is converted to a **coda** object meaning that all function for MCMC analysis available in the R package **coda** (Plummer *et al.* 2006) are available.

4. Other functionalities

4.1. Filtering

There are two functions related to filtering:

```
ht(object, theta, y)
Pstate(object, theta, y)
```

As in the `pdf` and `cdf` functions, the `object` argument accepts a specification created with `create.spec` (see Section 2) or a fitted object (see Section 3). When a specification is passed to the method, `theta` and `y` must be provided which is not the case when a fitted object is passed to `object`. The function `ht` outputs the filtered volatility of each regime by simply running the variance update function (see Section 2.1) of each single-regime specification up to time $T + 1$. The function `Pstate` runs the Hamilton filter (see Section 3) and gives the states probabilities up to time $T + 1$. These functions are useful to analyze the evolution of the volatilities and state probabilities over time.

4.2. Simulation

Simulations are carried out by two functions:

```
sim(spec, n, m, theta, burnin)
simahead(object, n, m, theta, y)
```

The `object` argument works like the `object` argument of the previous functions. The function `sim` simulate an entire process while the function `simahead` simulate `n`-step ahead to the vector of observation `y`. The argument `n` sets the length of each simulation while the argument `m` sets the number of simulations. If a matrix of parameters such as a MCMC chain (see Section 3.0.2) is passed to `theta`, we automatically sets `m = M` where `M` is the number of MCMC draws. Finally, the argument `burnin`, unique to `sim`, sets the number of simulation to discare to diminishes the impact of initial state probabilities set to $1/K$, where K is the number of states.

4.3. Predictive density and Probability integral transform

The predictive density function is essentially useful for a MCMC chain since it does not behave like the function `pdf` when given a matrix of parameters:

```
pred(object, x, theta, y, log = TRUE, do.its = FALSE)
pit(object, x, theta, y, do.norm, is.its = FALSE)
```

The difference between the `pdf` and the `pred` function is that when a matrix of parameters is passed to `theta`, each vector of parameters are not evaluated individually. That is, we average the resulting PDF of each density estimation of each individiual vector of parameters in the matrix.

$$f(x) \equiv \frac{1}{M}\sum_{m=1}^{M} f(x|H_{T+1}^{[m]}, \psi_{T+1}^{[m]}, \Lambda^{[m]})\,. \tag{18}$$

where M is the total number of draws in the MCMC chain. When a single vector of parameters is passed to `theta` the results are the same as calling the function `pdf` (see Section(2.3)). When `do.its = TRUE`, we calculate the in-sample predictive evaluated at the observations in `y`. As for the Probability intgral transform (PIT), only replace `pdf` by `cdf`. The `do.norm` argument serves as to transform the PIT into standard Normal variate.

4.4. Risk measures

Calculation of the Value-at-Risk (VaR) and Expected-shortfall (ES) is a crucial step in risk management. We provide the function `risk` which allows us to calculate these risk measures in-sample or at time $t = T + 1$.

```
risk(object, theta, y, level = 0.95, ES = FALSE, do.its = FALSE)
```

The `object` argument works as in the previous functions. The argument `level` is a vector that sets the risk level. The argument `ES` indicates if the ES is computed. Finally, the argument `do.its` indicates if in-sample evaluation or $T + 1$ evaluation is performed.

The function can calculate the risk estimators from Maximum likelihood or Bayesian estimation. We first extract the predictive density $f(x)$ given the information at time t. This is essentially the `pred` function with `do.its = FALSE`. When `do.its = TRUE` in the `risk` function, we iteratively pass `y` up to observation t in the function `pred` which gives us the predictive density at time $t+1$. We do this up to time $T-1$ which gives us the risk estimators from $t = 2$ to $t = T$. When `do.its = FALSE`, we pass all data in `y` and obtain the risk estimators at time $T + 1$. To extract the VaR, we solve the VaR at risk level $1 - \alpha$:

$$\int_{-\infty}^{VaR_\alpha} f(x)dx = \alpha\,. \tag{19}$$

We do this by first running the R function `uniroot` from the `stats` namespace for a gross approximate of the VaR and then we run the Newton-Raphson algorithm to obtain better estimations. The ES is then found by integration:

$$ES_\alpha \equiv \int_{-\infty}^{VaR_\alpha} x f(x)dx\,. \tag{20}$$

We use the R function `integrate` from the `stats` namespace to perform the numerical integration.

4.5. Information critera

The Akaike information criterion (AIC) (Akaike 1974), the Bayesian information criterion (BIC) (Schwarz *et al.* 1978), and the deviance information criterion (DIC) (Gelman *et al.* 2014) are all measure of the relative quality of statistical models for a given set of data where lower measure are preferred.

```
AIC(fit)
BIC(fit)
DIC(fit)
```

Both BIC and AIC prevents overfitting by introducing a penalty term to the number of parameters k in the model, but the penalty term is larger in the BIC than in the AIC. When a Bayesian fit is passed to the `BIC` or `AIC` function, the the BIC or AIC on the posterior mean $\bar{\Theta}$,$\bar{\Lambda}$, and $\bar{\mathbf{P}}$ is calculated.

The DIC is particularly useful in Bayesian model selection problems where the posterior distributions of the models have been obtained by MCMC simulation (see Section 3). We define the deviance as $D(\Theta, \Lambda, \mathbf{P})) \equiv -2LLH(\mathbf{y}|\Theta, \Lambda, \mathbf{P})$, where y are the data. The expectation $\bar{D} \equiv \mathbf{E}^{\Theta,\Lambda,\mathbf{P}}[D(\Theta, \Lambda, \mathbf{P})]$ is a measure of how well the model fits the data. The larger this is, the worse the fit. The effective number of parameters of the model can be define as $p_D \equiv \frac{1}{2}\widehat{\text{var}}\left(D(\Theta, \Lambda, \mathbf{P})\right)$. The larger the effective number of parameters is, the easier it is for the model to fit the data, and so the deviance needs to be penalized.

5. Empirical illustration

We illustrate the package's usage on daily log-returns of the Swiss market index for a period ranging from November 12, 1990, to October 20, 2000. We are using the data from Mullen *et al.* (2011). We consider a single-regime GJR model with a skewed Student-t distribution and a two-state Markov-switching GJR model with skewed Student-t distributions in each regime. Figure 1 displays the time series of log-returns.

[Insert Figure 1 about here.]

We first estimate both models by Maximum Likelihood with the pre-optimization argument `do.init = TRUE` and the argument `do.enhance.theta0 = TRUE`:

```
R> data("SMI")
R> plot(y, xlab = "Date", ylab = "Log-return")
R> SMI = as.matrix(y)
R> date = as.Date(rownames(SMI))
R> date = c(date, date[length(date)] + 1)
R> spec1 = create.spec(model = c("gjrGARCH"),
                       distribution = c("std"),
                       do.skew = c(TRUE),
                       do.mix = FALSE, do.shape.ind = FALSE)
R> ctr.mle1 = list(do.init = TRUE, NP = 10*length(spec1$theta0),
                   itermax = 500, do.enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle1 = MSGARCH::fit.mle(spec = spec1, y = SMI, ctr = ctr.mle1)
R> summary(fit.mle1)
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: gjrGARCH_student_skew"
```

```
[1] "Number of parameters in variance model: 4"
[1] "Number of parameters in distribution: 2"
[1] "Default parameters:"
     alpha0_1 alpha1_1 alpha2_1 beta_1 nu_1 xi_1
[1,]      0.1     0.05      0.1    0.8   10    1
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
     alpha0_1 alpha1_1 alpha2_1 beta_1  nu_1
[1,]  0.03933  0.04298   0.1143 0.8702 8.138
       xi_1
[1,] 0.8554
[1] "Unconditional volatility:"
     State 1
[1,]   1.285
[1] "Log-kernel: -3376.27190288128"
[1] "AIC: 6743.30840993793"
[1] "BIC: 6778.25268600307"
```

```
R> ht = MSGARCH::ht(fit.mle1)
R> plot(ht, date = date)
```

[Insert Figures 2 about here.]

We plot in Figure 2 the conditional variance of the single-regime model. We note the high level of volatility persistence, skewness and a fat-tailed distribution.

```
R> spec2 = create.spec(model = c("gjrGARCH", "gjrGARCH"),
                       distribution = c("std", "std"),
                       do.skew = c(TRUE, TRUE),
                       do.mix = FALSE, do.shape.ind = FALSE)
R> ctr.mle2 = list(do.init = TRUE, NP = 50*length(spec2$theta0),
                   itermax = 500, do.enhance.theta0 = TRUE)

R> set.seed(123)
R> fit.mle2 = MSGARCH::fit.mle(spec = spec2, y = SMI, ctr = ctr.mle2)
R> summary(fit.mle2)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: gjrGARCH_student_skew gjrGARCH_student_skew"
[1] "Number of parameters in each variance model: 4 4"
[1] "Number of parameters in each distribution: 2 2"
[1] "Default parameters:"
     alpha0_1 alpha1_1 alpha2_1 beta_1 nu_1 xi_1
[1,]      0.1     0.05      0.1    0.8   10    1
     alpha0_2 alpha1_2 alpha2_2 beta_2 nu_2 xi_2
[1,]      0.1     0.05      0.1    0.8   10    1
       P   P
```

```
[1,] 0.5 0.5
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
     alpha0_1  alpha1_1 alpha2_1 beta_1  nu_1
[1,]   0.2229 7.648e-06   0.2139 0.5404 5.945
       xi_1 alpha0_2 alpha1_2 alpha2_2 beta_2
[1,] 0.8521    0.083 0.006211   0.1393 0.8774
      nu_2   xi_2      P       P
[1,] 20.01 0.8582 0.9981 0.00313
[1] "Transition matrix:"
             t = 1   t = 2
t + 1 = 1 0.998052 0.00313
t + 1 = 2 0.001948 0.99687
[1] "Stable probabilities:"
        Stable probabilities
State 1               0.5464
State 2               0.4536
[1] "Unconditional volatility:"
     State 1 State 2
[1,]  0.8101   1.423
[1] "Log-kernel: -3364.58904570219"
[1] "AIC: 6687.68084240918"
[1] "BIC: 6769.21748656117"
```

```
R> ht = ht(fit.mle2)
R> plot(ht, date = date)
R> state = Pstate(fit.mle2)
R> plot(state, date = date)
```

We first note that the first regime of the MSGARCH specification is less persitent then that of the second regime (see Figure 3). We can also observe that the `alpha2_1` parameter is very high showing a larger leverage effect inthe first regime than in the second regime. The estimated degrees of freedom suggests that the first regime is more fat-tailed than the second regime, but the unconditional volatility of the first regime is much lower than that of the second regime. Both conditional distributions are negatively skewed. As we could expect, the unconditional volatility of the single-regime specification is close to the average of the two unconditional volatilities of the MSGARCH specification. The transtion matrix indicates that the regime does not switch very often which can be seen in Figure 4.

[Insert Figure 3 and Figure 4 about here.]

We also perform the Bayesian estimation using as starting values the ML values.

```
R> ctr.bay1 = list(N.burn = 5000, N.mcmc = 10000,
                   N.thin = 10, theta0fit.mle1$theta)
R> set.seed(123)
```

```
R> fit.bay1 = MSGARCH::fit.bayes(spec = spec1, y = SMI, ctr = ctr.bay1)

R> summary(fit.bay1)

[1] "Specification Type: Single-Regime"
[1] "Specification Name: gjrGARCH_student_skew"
[1] "Number of parameters in variance model: 4"
[1] "Number of parameters in distribution: 2"
[1] "Default parameters:"
     alpha0_1 alpha1_1 alpha2_1 beta_1 nu_1 xi_1
[1,]      0.1     0.05      0.1    0.8   10    1
[1] "Bayesian posterior mean:"
     alpha0_1 alpha1_1 alpha2_1   beta_1     nu_1
     0.017720 0.013427 0.009124 0.981166 3.402079
         xi_1
     1.027325
[1] "Posterior variance-covariance matrix"
           alpha0_1   alpha1_1   alpha2_1
alpha0_1  4.531e-05  3.160e-06  1.520e-05
alpha1_1  3.160e-06  6.317e-06 -4.729e-06
alpha2_1  1.520e-05 -4.729e-06  2.238e-05
beta_1   -1.352e-05 -4.002e-06 -6.982e-06
nu_1      6.904e-06 -1.256e-05  3.064e-05
xi_1      1.325e-06  7.229e-06 -1.804e-05
             beta_1       nu_1       xi_1
alpha0_1 -1.352e-05  6.904e-06  1.325e-06
alpha1_1 -4.002e-06 -1.256e-05  7.229e-06
alpha2_1 -6.982e-06  3.064e-05 -1.804e-05
beta_1    8.097e-06 -2.742e-06  1.338e-06
nu_1     -2.742e-06  5.052e-05 -3.088e-05
xi_1      1.338e-06 -3.088e-05  2.019e-05
[1] "Posterior mean unconditional volatility:"
     State 1
[1,]   4.229
[1] "Acceptance rate: 0.992"
[1] "AIC: 22471.2046441839"
[1] "BIC: 22509.7141828453"
[1] "DIC: 22468.3407630412"

R> coda::traceplot(fit.bay1$theta)
R> pairs(x = as.matrix(fit.bay1$theta), pch = 20, cex = 0.8)
R> ht = ht(fit.bay1)
R> plot(ht, date = date)
```

[Insert Figure 5 and Figure 6 about here.]

From Figure 5 we observe that the chain is mixing well. Figure 6 shows that `nu_1` and `alpha2_1` are highly positively correlated while `beta_1` and `alpha0_1` are highly negatively correlated.

[Insert Figure 7 about here.]

The Bayesian conditional variance in Figure 7 looks similar to the MLE case, but now with a narrow band indicating relatively low uncertainty.

```
R> ctr.bay2 = list(N.burn = 5000, N.mcmc = 10000,
                   N.thin = 10, theta0 = fit.mle2$theta)
R> set.seed(123)
R> fit.bay2 = MSGARCH::fit.bayes(spec = spec2, y = SMI, ctr = ctr.bay2)

R> summary(fit.bay2)
```

```
[1] "Specification Type: Markov-Switching"
[1] "Specification Name: gjrGARCH_student_skew gjrGARCH_student_skew"
[1] "Number of parameters in each variance model: 4 4"
[1] "Number of parameters in each distribution: 2 2"
[1] "Default parameters:"
     alpha0_1 alpha1_1 alpha2_1 beta_1 nu_1 xi_1
[1,]      0.1     0.05      0.1    0.8   10    1
     alpha0_2 alpha1_2 alpha2_2 beta_2 nu_2 xi_2
[1,]      0.1     0.05      0.1    0.8   10    1
       P   P
[1,] 0.5 0.5
[1] "Bayesian posterior mean:"
alpha0_1  alpha1_1  alpha2_1    beta_1      nu_1
0.223706  0.006811  0.234363  0.535547  5.951993
    xi_1  alpha0_2  alpha1_2  alpha2_2    beta_2
0.848133  0.084816  0.011409  0.155874  0.866230
     nu_2      xi_2         P         P
19.997764  0.862992  0.996872  0.004549
[1] "Posterior variance-covariance matrix"
           alpha0_1   alpha1_1   alpha2_1
alpha0_1  8.391e-05 -4.415e-06 -1.010e-04
alpha1_1 -4.415e-06  2.955e-05  8.658e-05
             beta_1       nu_1       xi_1
alpha0_1  2.623e-05 -2.188e-04  1.144e-04
alpha1_1 -7.793e-06  4.085e-05 -2.881e-05
           alpha0_2   alpha1_2   alpha2_2
alpha0_1 -6.799e-05  6.192e-06 -1.172e-04
alpha1_1  3.302e-05  2.371e-06  7.230e-05
             beta_2       nu_2       xi_2
alpha0_1  8.185e-05  8.469e-06  5.016e-05
alpha1_1 -4.671e-05 -6.221e-05  1.984e-05
                  P          P
```

```
alpha0_1 -4.778e-07 -1.360e-06
alpha1_1  2.629e-07  8.527e-07
[ reached getOption("max.print") -- omitted 12 rows ]
[1] "Posterior mean transition matrix:"
             t = 1    t = 2
t + 1 = 1 0.996872 0.004549
t + 1 = 2 0.003128 0.995451
[1] "Posterior mean stable probabilities:"
        Stable probabilities
State 1               0.5497
State 2               0.4503
[1] "Posterior mean unconditional volatility:"
      State 1 State 2
[1,]  0.8286   1.488
[1] "Acceptance rate: 0.984"
[1] "AIC: 6689.39263252908"
[1] "BIC: 6770.92927668107"
[1] "DIC: 6672.09065870322"
```

```
R> coda::traceplot(fit.bay2$theta)
R> pairs(x = as.matrix(fit.bay2$theta[,c(1,3,4,7,9,10)]),
          pch = 20, cex = 0.8)
R> ht = ht(fit.bay2)
R> plot(ht, date = date)
R> state = Pstate(fit.bay2)
R> plot(state, date = date)
```

[Insert Figure 8 and Figure 9 about here.]

From Figure 8 we see that the chain is also mixing well fot the MSGARCH model. We can observe in Figure 9 that there is an high positive correlation between `alpha2_1` and `alpha2_2`, an high negative correlation between `alpha2_1` and `beta_2`, and an high negative correlation between `alpha2_2` and `beta_2`.

[Insert Figure 10 and Figure 11 about here.]

As in the single regime model, the Bayesian conditional volatility is similar the MLE counterpart. We can obsvere that the band on the second regime is larger than the one of the first regime indicating more uncertainty in the second regime. The same effect is observe in Figure 11 where we show the probability to be in the first state.

To determine the best model in-sample, we compare the AIC and the BIC from the ML esimation and the DIC from the Bayesian esimation. Lowest values should be preferred:

```
R> c(MSGARCH::AIC(fit.mle1), MSGARCH::AIC(fit.mle2))
```

```
[1] 6743 6688
```

```
R> c(MSGARCH::BIC(fit.mle1), MSGARCH::BIC(fit.mle2))
```

```
[1] 6778 6769
```

```
R> c(MSGARCH::DIC(fit.bay1)$DIC, MSGARCH::DIC(fit.bay2)$DIC)
```

```
[1] 6740 6672
```

We can observe that the MSGARCH specification is better for all information criteria.

We now analyze the Value-at-Risk at the 95% risk level for all models:

```
R> risk.mle1 = MSGARCH::risk(fit.mle1, level = c(0.95),
                             ES = FALSE, do.its = TRUE)
R> risk.mle2 = MSGARCH::risk(fit.mle2, level = c(0.95),
                             ES = FALSE, do.its = TRUE)
R> risk.bay1 = MSGARCH::risk(fit.bay1, level = c(0.95),
                             ES = FALSE, do.its = TRUE)
R> risk.bay2 = MSGARCH::risk(fit.bay2, level = c(0.95),
                             ES = FALSE, do.its = TRUE)
```

```
R> par(oma = c(4, 1, 1, 1))
R> plot(zoo::zoo(risk, order.by = date),plot.type = "single",
                 col = tsRainbow, ylab = "VaR",xlab = "Date")
R> legend("bottomright",legend =  colnames(risk),
          lty = 3, col = tsRainbow, xpd = TRUE, horiz = TRUE,
R> inset = c(0,-0.5), bty = "n", pch = c(4, 2, 15, 19), cex = 1)
```

[Insert Figure 12 about here.]

The Value-at-Risk at 5% for both MLE estimation of each model can be seen in Figure 12. They look similar except that the MSGARCH model often shows bigger spikes than the single-regime model when there is a large shift in volatility.

6. Conclusion

This vignette introduced the R package **MSGARCH** which allows us to estimate, simulate and forecast Markov-switching GARCH models in the R statistical sofware. We detailed how to create various single-regime and regime-switching specifications with various scedastic functions and conditional distributions. We documented how to perfom Maximum Likelihood and Bayesian estimation of these models. In an empirical illustration to real financial data, we showed how to fit and compare the in-sample performance of two complex GARCH specifications.

The R language has become an important vector for knowledge transfer in quantitative finance over the last years. We hope the R package **MSGARCH** will provide risk managers and regulators with new methodologies for improving risk forecasts of their portfolios.

Finally, if you use R or **MSGARCH**, please cite the software in publications.

Computational details

The results in this paper were obtained using R 3.2.3 (R Core Team 2016) with the packages: **MSGARCH** version 0.1.0 (Bluteau *et al.* 2016), **adaptMCMC** version XXX (Andreas 2012), **DEoptim** version XXX (Mullen *et al.* 2011), **nloptr** version XXX (Johnson 2014), **Rcpp** version 0.12.5 (Eddelbuettel and François 2011; Eddelbuettel *et al.* 2016a), **RcppArmadillo** version 0.7.100.3.1 (Eddelbuettel and Sanderson 2014; Eddelbuettel *et al.* 2016b), **Rsolnp** version 1.15 (Ghalanos and Theussl 2016), **xts** version 0.9-7 (Ryan and Ulrich 2015) and **quantmod** version 0.4-5 (Ryan 2015). R itself and all packages used are available from CRAN at `http://CRAN.R-project.org/`. The package **MSGARCH** is under development in GitHub at `https://github.com/keblu/MSGARCH`. Computations were performed on a Genuine Intel® quad core CPU i7–3630QM 2.40Ghz processor. Code outputs were obtained using `options(digits = 4, max.print = 40, prompt = "R> ", width = 50)`.

Acknowledgments

The authors acknowledge Google for financial support via the Google Summer of Code 2016 project ”MSGARCH”; see `https://summerofcode.withgoogle.com/projects/?sp-search=KBlutea#6497774455488512`, the International Institute of Forecasters (IIF, `https://forecasters.org`) and Industrielle-Alliance. We thank Denis-Alexandre Trottier for the development of a previous version of the package. Any remaining errors or shortcomings are the authors' responsibility.

References

Akaike H (1974). “A New Look at the Statistical Model Identification.” *Automatic Control, IEEE Transactions on*, **19**(6), 716–723. `doi:10.1007/978-1-4612-1694-0_16`.

Andreas S (2012). ***adaptMCMC**: Implementation of a Generic Adaptive Monte Carlo Markov Chain Sampler.* URL `https://cran.r-project.org/package=adaptMCMC`.

Ardia D (2008). *Financial Risk Management with Bayesian Estimation of GARCH Models: Theory and Applications*, volume 612 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin. `doi:10.1007/978-3-540-78657-3`. ISBN 978-3-540-78656-6.

Black F (1976). “Studies of Stock Price Volatility Changes.” In *Proceedings of the 1976 Meetings of the Business and Economics Statistics Section*, pp. 177–181. American Statistical Association.

Bluteau K, Ardia D, Trottier DA, Bout K, Peterson B (2016). ***MSGARCH**: Markov-Switching GARCH Models in R*. R package version 0.1.0, URL `https://github.com/LeopoldoCatania/GAS`.

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

Caporale GM, Pittis N, Spagnolo N (2003). "IGARCH Models and Structural Breaks." *Applied Economics Letters*, **10**(12), 765–768. doi:10.1080/1350485032000138403.

Christie AA (1982). "The Stochastic Behavior of Common Stock Variances: Value, Leverage and Interest Rate Effects." *Journal of financial Economics*, **10**(4), 407–432. doi:10.1016/0304-405x(82)90018-6.

Creal D, Koopman SJ, Lucas A (2013). "Generalized Autoregressive Score Models with Applications." *Journal of Applied Econometrics*, **28**(5), 777–795. doi:10.1002/jae.1279.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, François R, Allaire J, Ushey K, Kou Q, Bates D, Chambers J (2016a). ***Rcpp**: Seamless R and C++ Integration.* R package version 0.12.5, URL https://cran.r-project.org/package=Rcpp.

Eddelbuettel D, François R, Bates D (2016b). ***RcppArmadillo**: **Rcpp** Integration for the **Armadillo** Templated Linear Algebra Library.* R package version 0.7.100.3.1, URL https://cran.r-project.org/package=RcppArmadillo.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High–Performance C++ Linear Algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Fernández C, Steel MF (1998). "On Bayesian Modeling of Fat Tails and Skewness." *Journal of the American Statistical Association*, **93**(441), 359–371. doi:10.1080/01621459.1998.10474117.

Francq C, Zakoian JM (2011). *GARCH models: Structure, statistical inference and financial applications.* John Wiley & Sons. doi:10.1002/9780470670057.

Gelman A, Carlin JB, Stern HS, Rubin DB (2014). *Bayesian Data Analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA. doi:10.2307/2965436.

Ghalanos A, Theussl S (2016). ***Rsolnp**: General Non–Linear Optimization using Augmented Lagrange Multiplier Method.* R package version 1.16, URL https://cran.r-project.org/package=Rsolnp.

Glosten LR, Jagannathan R, Runkle DE (1993). "On the Relation Between the Expected Value and the Volatility of the Nominal Excess Return on Stocks." *The Journal of Finance*, **48**(5), 1779–1801. doi:10.1111/j.1540-6261.1993.tb05128.x.

Haas M, Mittnik S, Paolella MS (2004a). "A New Approach to Markov-Switching GARCH Models." *Journal of Financial Econometrics*, **2**(4), 493–530. doi:10.1093/jjfinec/nbh020.

Haas M, Mittnik S, Paolella MS (2004b). "Mixed Normal Conditional Heteroskedasticity." *Journal of Financial Econometrics*, **2**(2), 211–250. doi:10.1093/jjfinec/nbh009.

Hamilton JD (1989). "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle." *Econometrica*, pp. 357–384. doi:10.2307/1912559.

Hillebrand E (2005). "Neglecting Parameter Changes in GARCH Models." *Journal of Econometrics*, **129**(1), 121–138. doi:10.1016/j.jeconom.2004.09.005.

Johnson SG (2014). *The **NLopt** Nonlinear-Optimization Package.* URL https://cran.r-project.org/web/packages/nloptr/.

Kraft D (1988). *A Software Package for Sequential Quadratic Programming.* Wiss. Berichtswesen d. DFVLR.

Lamoureux CG, Lastrapes WD (1990). "Persistence in Variance, Structural Change, and the GARCH Model." *Journal of Business and Economic Statistics*, **8**(2), 225–234. doi:10.2307/1391985.

Mikosch T, Stărică C (2004). "Nonstationarities in Financial Time Series, the Long-Range Dependence, and the IGARCH Effects." *Review of Economics and Statistics*, **86**(1), 378–390. doi:10.1162/003465304323023886.

Mullen KM, Ardia D, Gil DL, Windover D, Cline J, *et al.* (2011). "**DEoptim**: An R Package for Global Optimization by Differential Evolution." *Journal of Statistical Software*, **40**(6), 1–26. doi:10.18637/jss.v040.i06.

Nelson DB (1991). "Conditional Heteroskedasticity in Asset Returns: A New Approach." *Econometrica*, pp. 347–370. doi:10.2307/2938260.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://cran.r-project.org/package=coda.

Price K, Storn RM, Lampinen JA (2006). *Differential Evolution: A Practical Approach to Global Optimization.* Springer Science & Business Media.

R Core Team (2016). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. R version 3.2.3, URL https://www.R-project.org/.

Ryan JA (2015). ***quantmod**: Quantitative Financial Modelling Framework.* R package version 0.4-5, URL https://CRAN.R-project.org/package=quantmod.

Ryan JA, Ulrich JM (2015). ***xts**: Extensible Time Series.* R package version 0.9-7, URL https://CRAN.R-project.org/package=xts.

Schwarz G, *et al.* (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Trottier DA, Ardia D (2016). "Moments of Standardized Fernandez-Steel Skewed Distributions: Applications to the Estimation of GARCH-Type Models." doi:10.1016/j.frl.2016.05.006. Forthcoming in Finance Research Letters.

Vihola M (2012). "Robust Adaptive Metropolis Algorithm with Coerced Acceptance Rate." *Statistics and Computing*, **22**(5), 997–1008. doi:10.1007/s11222-011-9269-5.

Zakoian JM (1994). "Threshold Heteroskedastic Models." *Journal of Economic Dynamics & Control*, **18**(5), 931–955. doi:10.1016/0165-1889(94)90039-6.

Figure 1: Log-returns of the Swiss Market Index. Data range from November 12, 1990, to October 20, 2000.

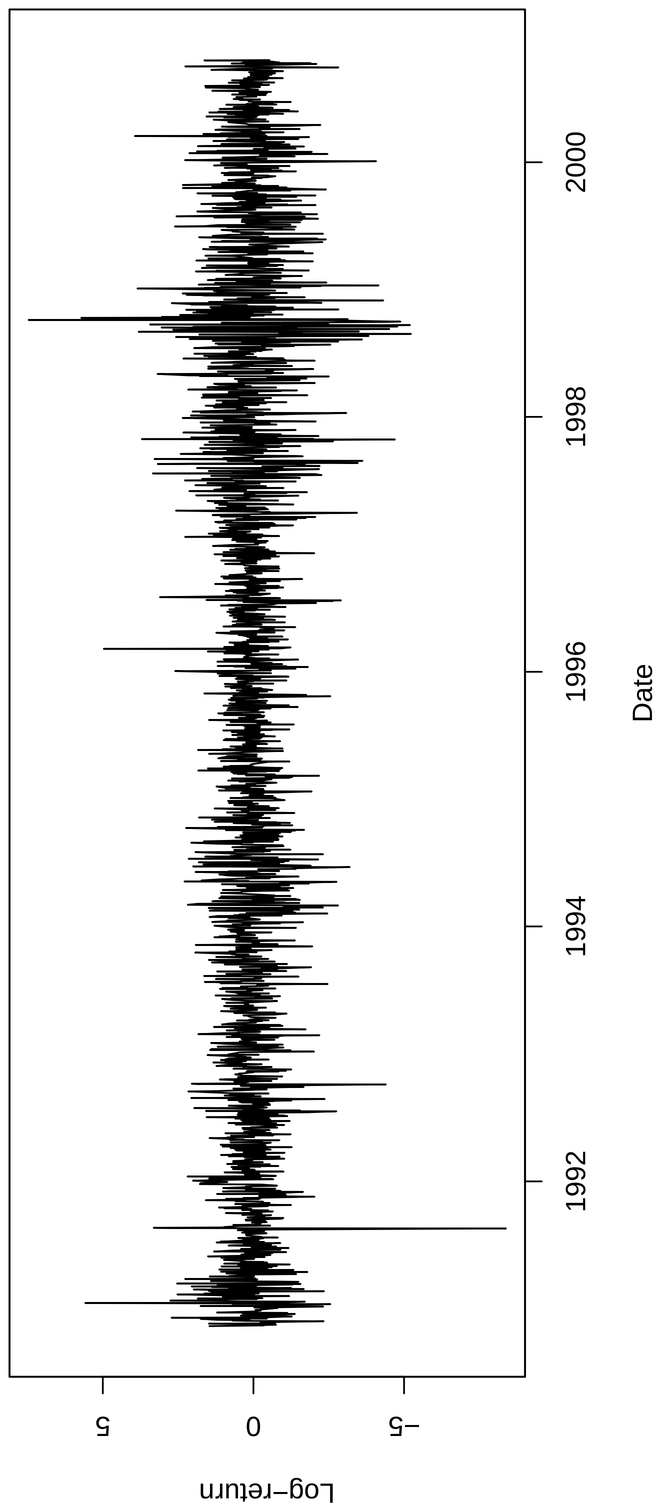


Figure 2: Conditional volatility of the single-regime GJR model with skewed Student-t innovations estimated by Maximum Likelihood.

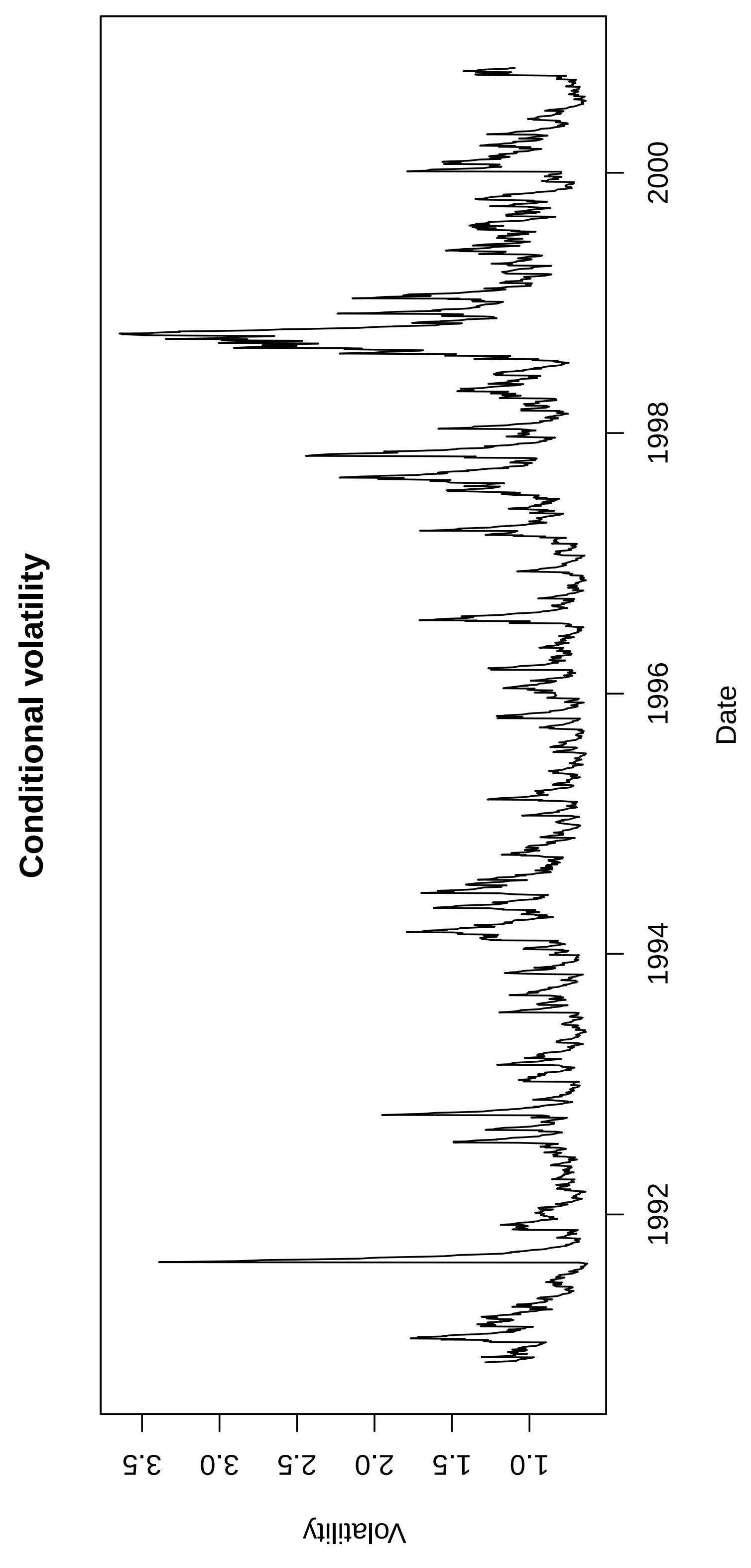
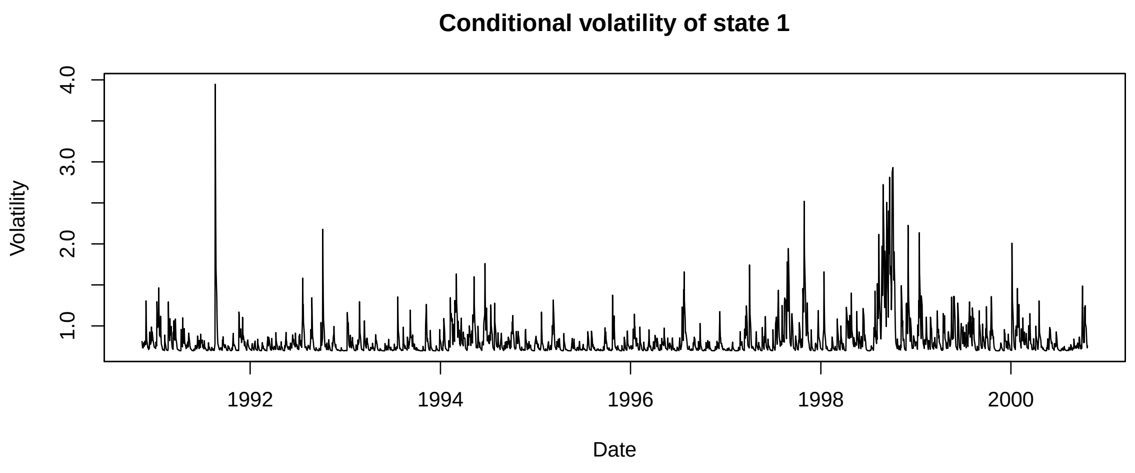


Figure 3: Conditional volatility in each states of the two-state Markov-switching GJR with skewed Student-t innovations estimated by Maximum Likelihood.



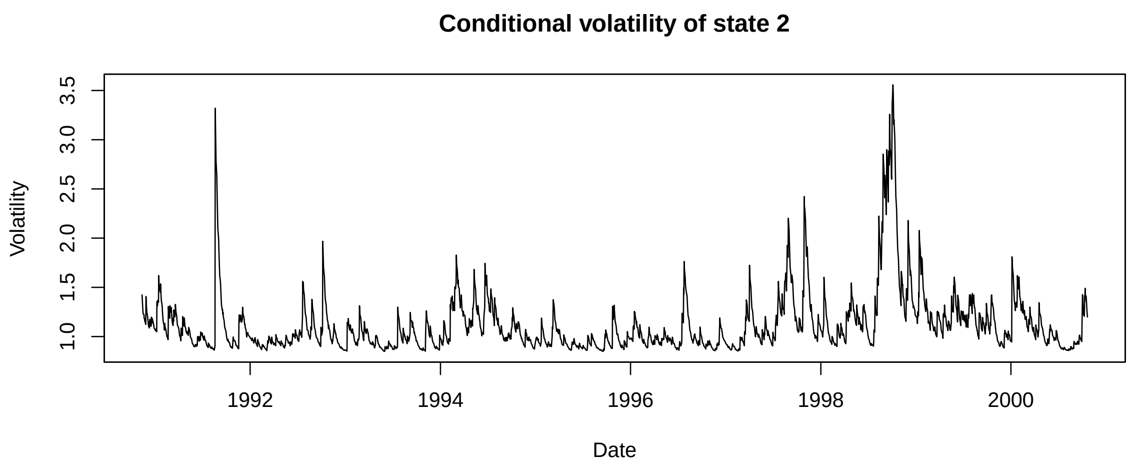


Figure 4: Filtered probabilities of the first regime obtain by Maximum Likelihood for the two-state Markov-switching GJR model with skewed Student-t innovations.

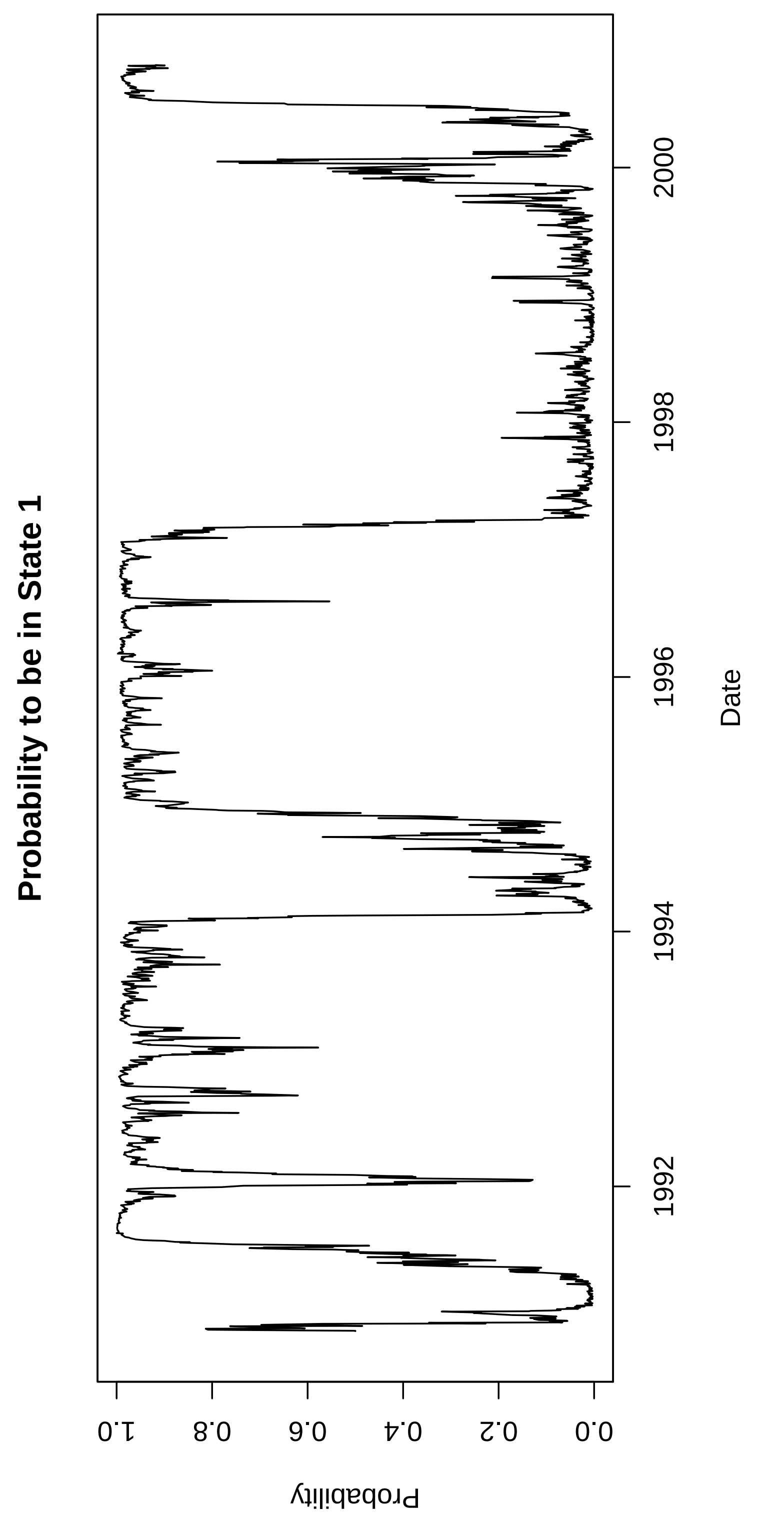


Figure 5: Trace of the single-regime GJR model with skewed Student-t innovations.

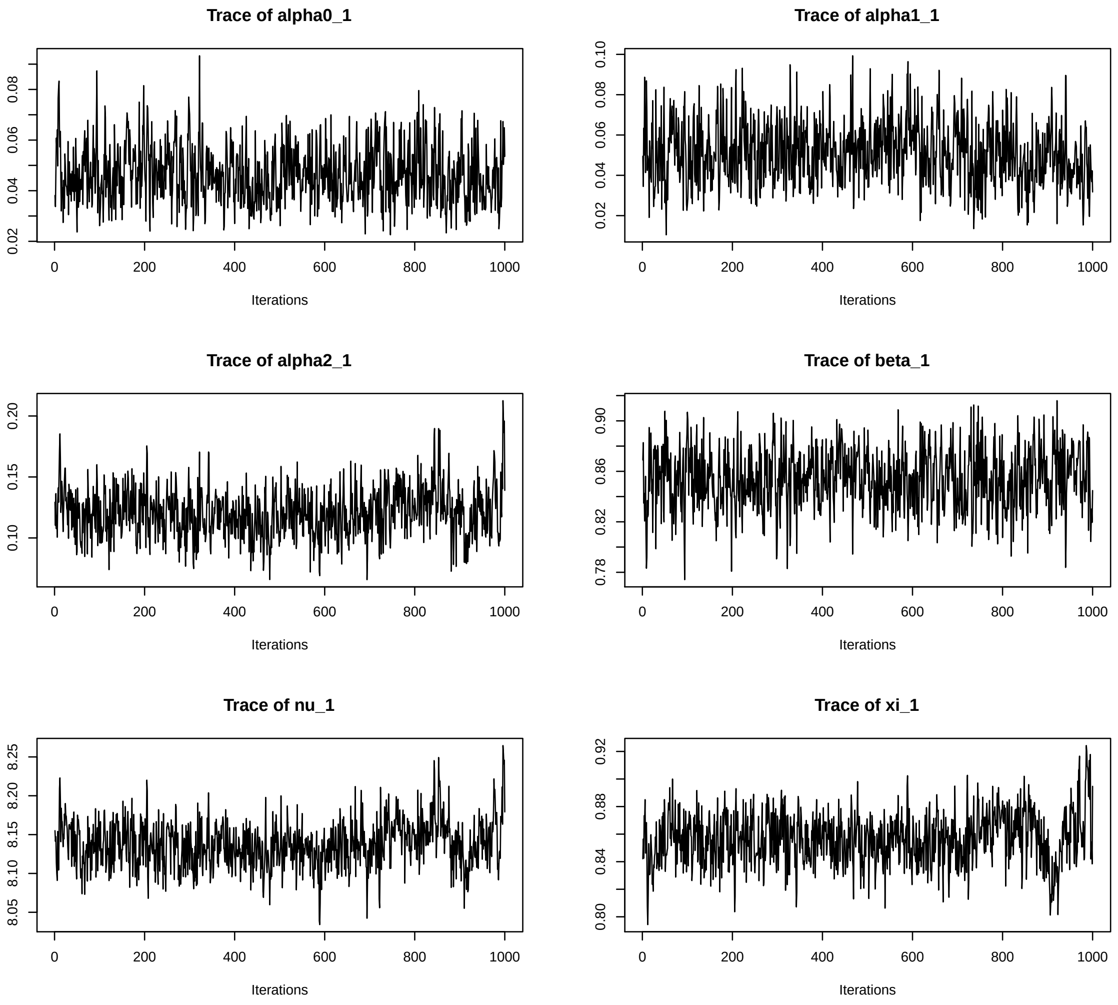


Figure 6: Pairs plot of the single-regime GJR model with skewed Student-t innovations.

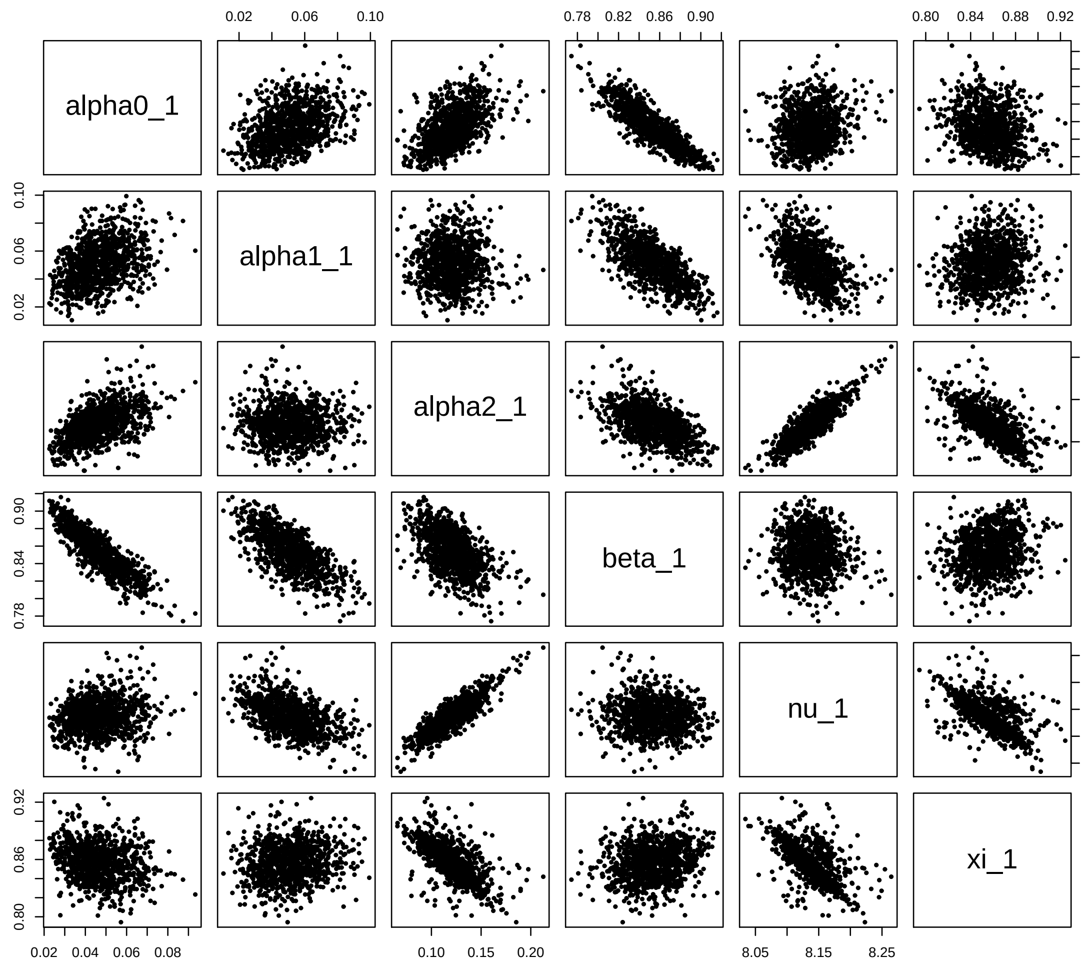


Figure 7: Conditional volatility of the single-regime GJR model with skewed Student-t innovations estimated by MCMC. Blue line indicates the median.

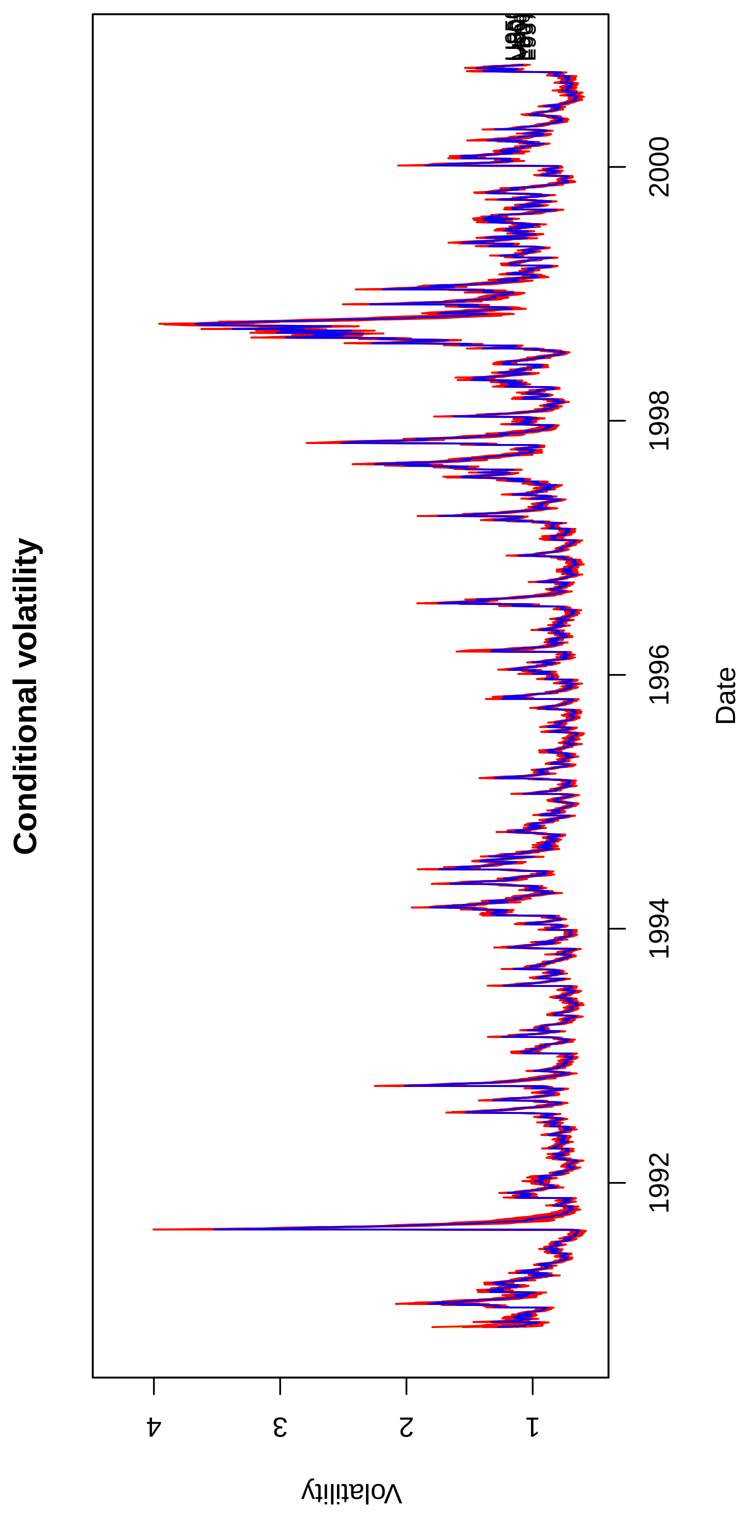


Figure 8: Trace of the two-state Markov-switching GJR with skewed Student-t innovations.

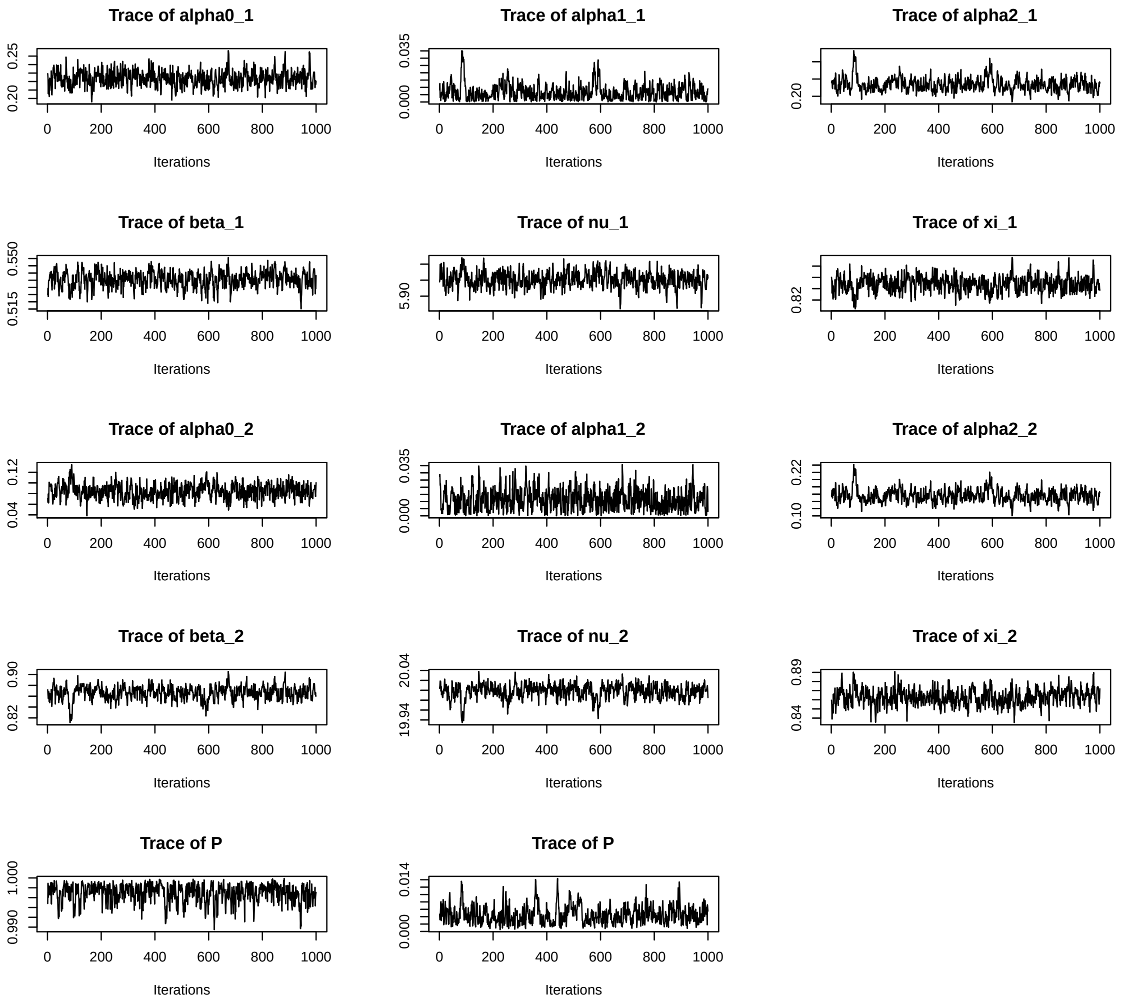


Figure 9: Pairs plot of the two-state Markov-switching GJR with skewed Student-*t* innovations.

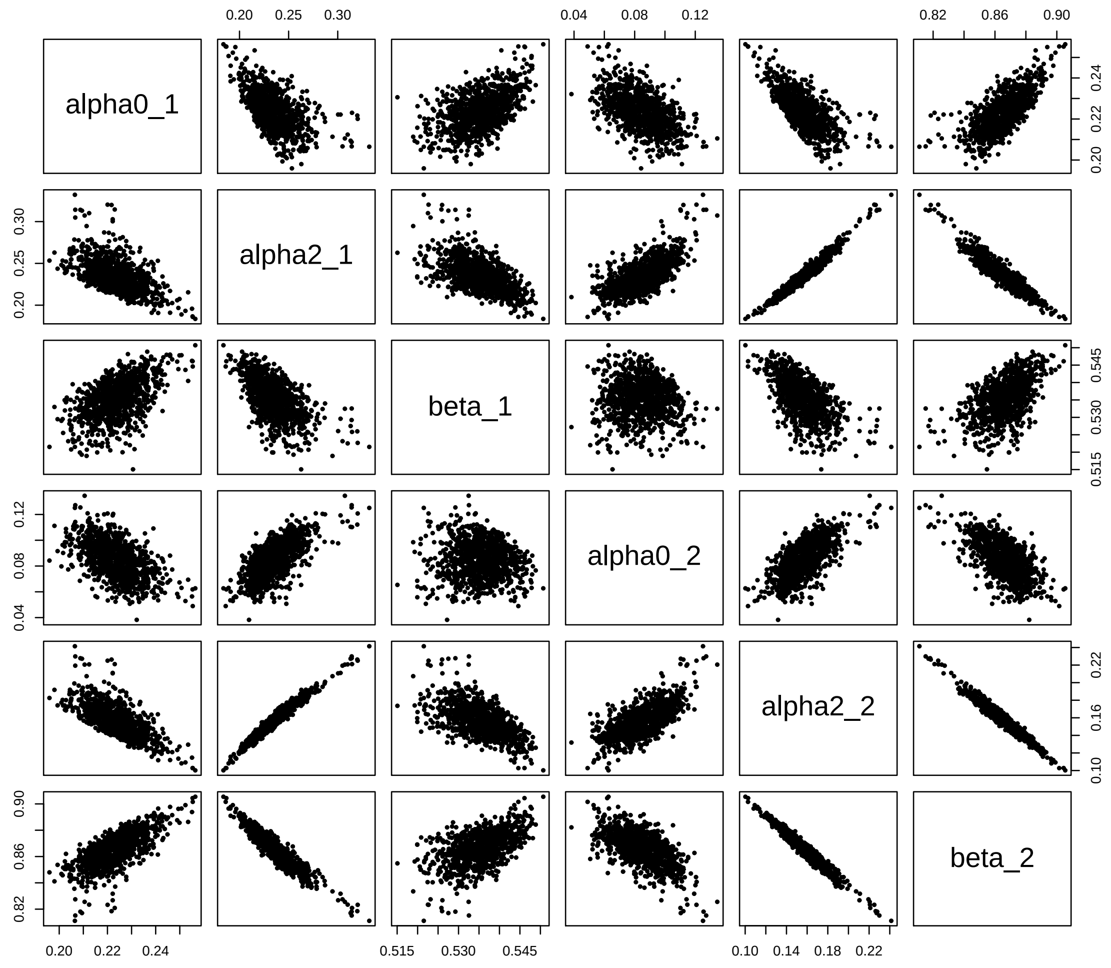
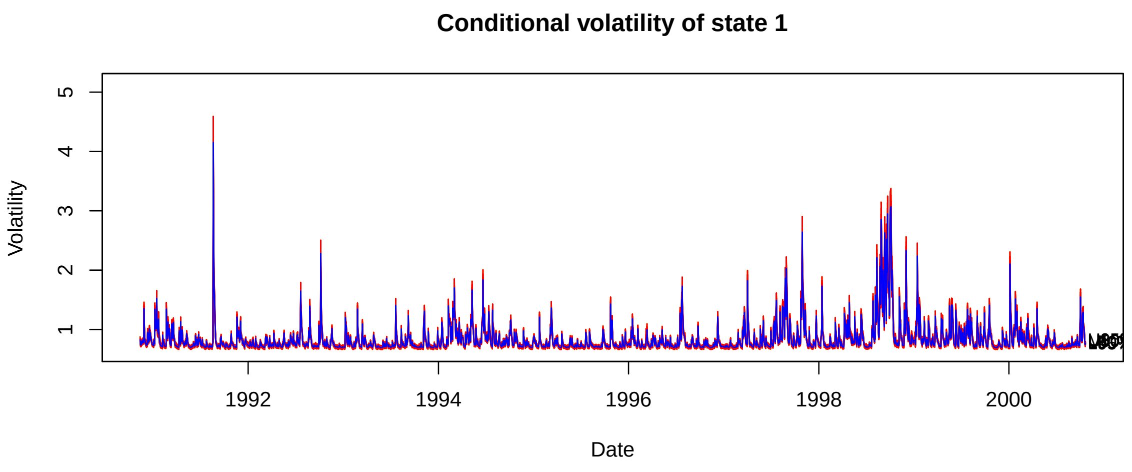


Figure 10: Conditional volatility in each states of the two-state Markov-switching GJR with skewed Student-t innovations estimated by MCMC. Blue line indicates the median.



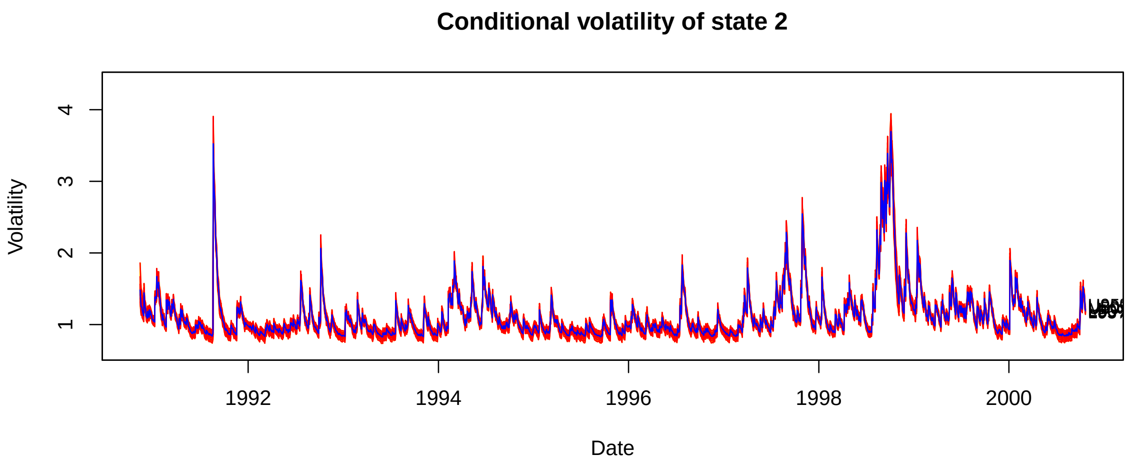


Figure 11: Filtered probabilities of the first regime obtain by MCMC for the two-state Markov-switching GJR model with skewed Student-t innovations. Blue line indicates the median.

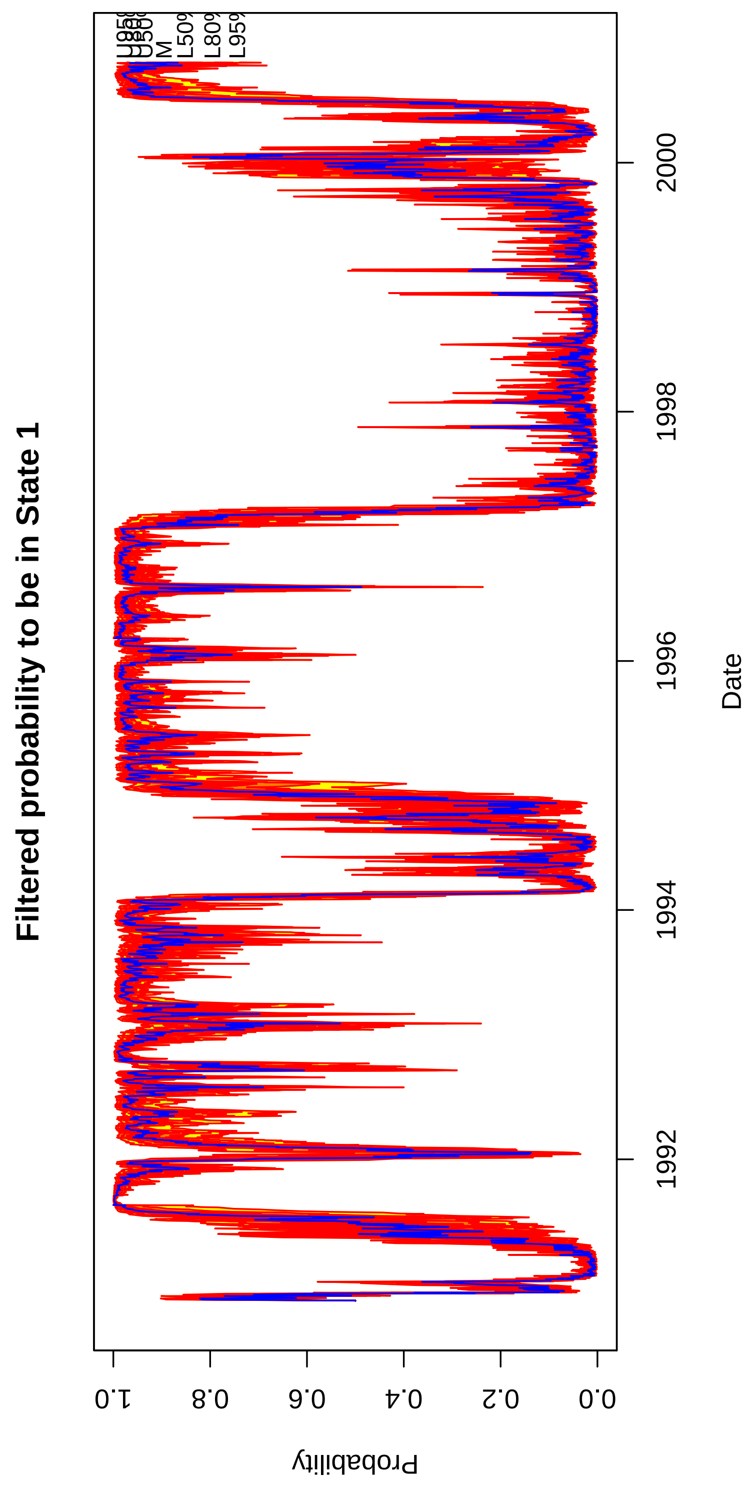
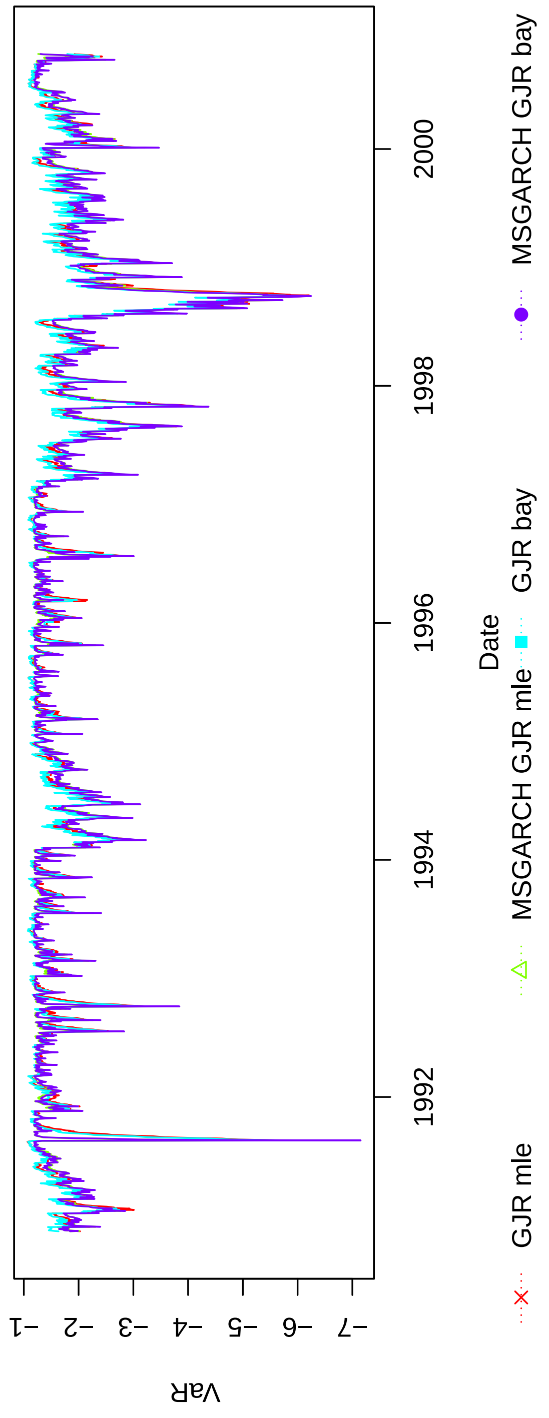


Figure 12: In-sample Value-at-Risk at the 95% risk leval for the single-regime and regime-switching models.



Affiliation:

David Ardia
Institute of Financial Analysis
University of Neuchâtel, Switzerland
and
Department of Finance, Insurance and Real Estate
Laval University, Canada
E-mail: `david.ardia@unine.ch`

Keven Bluteau (corresponding author)
Institute of Financial Analysis
Neuchatel University,Neuchatel, Switzerland
E-mail: `keven.bluteau@unine.ch`

Kris Boudt
Vrije Universiteit Brussel, Belgium
and
Vrije Universiteit Amsterdam, The Netherlands
E-mail: `kris.boudt@vub.ac.be`

Brian Peterson
E-mail: `brian@braverock.com`