Markov-Switching GARCH Models in R: The Package MSGARCH

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

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 risk management was the development of the GARCH model by Bollerslev (1986) where a time-varying conditional variance is given by a function of past returns and past conditional variance. The GARCH model is today a widespread tool for dealing with time series heteroscedastic models. However, estimation of the GARCH model often leads to parametrization that induces very high persistence or even non-stationarity (Caporale *et al.* 2003). Some possible explanations for this effect have been proposed in the financial literature. One of these proposals indicates that this effect might be induced by structural breaks in the volatility dynamics (Lamoureux and Lastrapes 1990; Mikosch and Stărică 2004; Hillebrand 2005). Markov-switching GARCH (MSGARCH) models such as the one proposed in Haas *et al.* (2004a) is designed to take structural breaks into account.

The R package **MSGARCH** aims to provide a comprehensive set of methods for MSGARCH processes, including fitting, filtering, forecasting, and simulating. Other methods such as Value-at-Risk, Expected-Shortfall, and distribution related functions are also available. In the R package **MSGARCH** there is no equation for the mean as in the R package **rugarch** (Ghalanos 2015). This means that we assume that before modeling, the user has filter the mean from their time series. In this document, we discuss the detail of the models and function available in this package. The document goes as follows: Section 2 shows the basic on how to create a specification, Section 3 presents the single-regime conditional variance processes available in the package, Section 4 presents the multiple-regime processes, Section 5 describes the available conditional distribution, Section Section 8 to Section 12 shows some of the package functionalities, and finally Section 13 demonstrates an application on real data.

2. Creating a specification

We provide in this section the step to create simple or complex specifications using the function `create.spec`. The simplest specification we can build is a GARCH model with a symmetric Normal conditional distribution:

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

The R package **MSGARCH** supports single-regime models as they are the building blocks for multiple-regime models. 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 3) 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 5) are `"norm"`, `"std"`, and `"ged"`. Each conditional distribution have a skewed version (see Section 5.4) 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 4.2) instead of a MSGARCH process (see Section 4.1). Finally, the the `do.shape.ind` argument allows for regime-independent shape parameters (see Section 4.3).

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 6 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
```

3. Single-regime specification

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

3.1. 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 $\mathcal{D}$ being 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 $\alpha_1 + \beta < 1$. The GARCH model captures volatility clustering which is a feature observed in financial data. When $\alpha_1 + \beta$ is near one, the volatility is highly persistent. The unconditional variance of the GARCH model is $\sigma = \alpha_0/(1 - \alpha_1 - \beta)$.

To create a single-regime GARCH specification we use `model = "sGARCH"` in the function `create.spec`.

```
R> create.spec(model = "sGARCH")
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: sGARCH_normal_sym"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
      alpha0 alpha1 beta
[1,]    0.1    0.1  0.8
```

3.2. 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 in 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 unconditional variance of the EGARCH model is $\sigma = \alpha_0/(1-\beta)$ (see Francq and Zakoian 2011, Section 10.1).

The creation of a single-regime EGARCH specification is done by using `model = "eGARCH"` in the function `create.spec`.

```
R> create.spec(model = "eGARCH")
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: eGARCH_normal_sym"
[1] "Number of parameters in variance model: 4"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
     alpha0 alpha1 alpha2 beta
[1,]   0.01    0.2   -0.1  0.8
```

3.3. 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 $\alpha_1 + \alpha_2 \mathsf{E}\left[\eta^2 \mathbb{I}_{\eta<0}\right] + \beta < 1$. Because of the term $\mathsf{E}\left[\eta^2 \mathbb{I}_{\eta<0}\right]$ in this expression, the GJR unconditional variance depends on the skewness of the distribution. The single-regime GJR specification is created by using `model = "gjrGARCH"` in the function `create.spec`:

```
R> create.spec(model = "gjrGARCH")
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: gjrGARCH_normal_sym"
[1] "Number of parameters in variance model: 4"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
     alpha0 alpha1 alpha2 beta
[1,]    0.1   0.05    0.1  0.8
```

3.4. 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 $\alpha_1^2 + \beta^2 - 2\beta(\alpha_1 + \alpha_2)\mathbb{E}\left[\eta \mathbb{I}_{\eta<0}\right] - (\alpha_1^2 - \alpha_2^2)\mathbb{E}\left[\eta^2 \mathbb{I}_{\eta<0}\right] < 1$ (see Francq and Zakoian 2011, Section 10.2). As in the GJR model, the unconditional variance is also dependent on the skewness of the distribution.

The single-regime TGARCH specification is created by using `model = "tGARCH"` in function `create.spec`:

```
R> create.spec(model = "tGARCH")

[1] "Specification Type: Single-Regime"
[1] "Specification Name: tGARCH_normal_sym"
[1] "Number of parameters in variance model: 4"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
     alpha0 alpha1 alpha2 beta
[1,]  0.125   0.05    0.1  0.8
```

3.5. 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:

$$h_t = \alpha_0 + \alpha_1 s_{t-1} + \beta h_{t-1}, \quad s_{t-1} = S_{t-1} \nabla_{t-1}, \quad \nabla_{t-1} = \left[\frac{\partial \ln f(y_{t-1} | h_{t-1}, \lambda)}{\partial h_{t-1}}\right],$$

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:

$$S_{t-1} \equiv \mathbb{E}[\nabla_{t-1} \nabla'_{t-1}]^{-1}$$

For positivity we set $\alpha_0 > 0$, $\alpha_1 \geq 0$ and $\beta \geq 0$. To ensure covariance-stationarity, we add the condition:

$$\sup_{h^* \in \mathcal{H}} \mathbb{E}\left[\left|\beta + \alpha_1 \left[\frac{\partial s(h^*)}{\partial h^*}\right]\right|\right] < 1$$

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

```
R> create.spec(model = "GAS")

[1] "Specification Type: Single-Regime"
[1] "Specification Name: GAS_normal_sym"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
     alpha0 alpha1 beta
[1,]    0.1    0.1  0.9
```

4. Multiple-regime specification

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

4.1. Markov-switching dynamics

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:

$$\mathbf{P} \equiv \begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{K,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,K} & p_{2,K} & \cdots & p_{K,K} \end{bmatrix}, \tag{6}$$

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 at time t be expressed as:

$$y_t \equiv \eta_{\Delta_t} h_{\Delta_t,t}^{1/2}, \tag{7}$$

where $\eta_{\Delta_t} \sim i.i.d.\mathcal{D}_{\Delta_t}(0, 1, \lambda_{\Delta_t})$ where $\mathcal{D}_{\Delta_t}$ is a distribution with zero mean, unit variance, and shape parameters λ_{Δ_t}. For a single-regime specification, we define θ as the parameters of the conditional variance process m, λ as the shape parameters of the conditional distribution $\mathcal{D}$, and the $(T \times 1)$ vector $\boldsymbol{h} \equiv (h_1, h_2, \ldots, h_T)'$ as the resulting conditional variance vector from this specification. A MSGARCH specification consists of many distinct single-regime specification linked by the transition matrix $\mathbf{P}$. Each single-regime conditional variance process $\boldsymbol{M} \equiv [m_1, m_2, ..., m_K]$ has their own parametrization $\Theta \equiv [\theta_1, \theta_2, ..., \theta_K]$ and each process follows a distinct conditional distribution $\boldsymbol{D} \equiv [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K]$ with shape parameters $\Lambda \equiv [\lambda_1, \lambda_2, ..., \lambda_K]$. This results in a $(T \times K)$ conditional variance matrix $\mathbf{H} \equiv [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_K]$.

To demonstrate the model, we create a two-states MSGARCH models with two single-regime GARCH processes each following a Normal distribution. We fit the model on the `sp500` dataset which consist of the S&P 500 index closing value log return from 1998-01-01 to 2015-12-31 from Yahoo Finance.

```
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 = 50*length(spec$theta0), itermax = 500,
                  enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> fit.mle$theta

        alpha0_1   alpha1_1   beta_1    alpha0_2   alpha1_2
[1,] 0.001598108 0.02710623 0.890987  0.03791289  0.1178041
          beta_2          P          P
     0.8776865 9.720447e-12 0.3684928

R> transmat.mle = MSGARCH::transmat(fit.mle)
R> transmat.mle

               [,1]      [,2]
1,]  9.720447e-12 0.3684928
[2,] 1.000000e+00 0.6315072
```

Here, we fit `sp500` on the created model by Maximum likelihood with the function `fit.mle` (see Section 6.1). The resulting parameters are store in the vector `theta` where each parameter are label according to the model and their state. The function `transmat` is an helper function that build the transition matrix from the fitted parameters for better readability of the last remaining parameters label as `P`.

4.2. Mixture of GARCH

Haas *et al.* (2004b) first proposed a general class of Mixture of GARCH models. They have developed, under their framework, a Mixture of Normal distribution where the variance process of each Normal component is a GARCH process. They named this specification the MNGARCH. 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. We can constrain the transition matrix $\mathbf{P}$ of the MSGARCH to make the probability $p_{i,j}$ of switching to state $\Delta_t = j$ from any state $\Delta_{t-1} = i$ the same. That is, $P(\Delta_t = j|\Delta_{t-1} = i) \equiv p_j$ $(i = 1, \ldots, K)$. The transition matrix is then reduce 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 of going to given state does not depend on the current state. For demonstration, lets repeat the experiment done previously, but with the argument `is.mix = TRUE`.

```
R> data("sp500ret")
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 = 50*length(spec$theta0), itermax = 500,
                  enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> fit.mle$theta

       alpha0_1  alpha1_1   beta_1   alpha0_2 alpha1_2
[1,] 0.00112834 0.0174685 0.897619 0.02928858 0.106416
        beta_2         P
     0.8883599 0.1954211

R> transmat.mle = MSGARCH::transmat(fit.mle)
R> transmat.mle

[1] 0.1954211 0.8045789
```

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 given that only the argument `is.mix` is changed. The `transmat` function, for a Mixture of GARCH, will output a probability vector and not a probability matrix.

4.3. Regime-independent shape distribution parameters

Sometimes it is useful to have a regime-switching behavior only in the conditional variance processes and have only one unique distribution. We call this regime-independent shape distribution parameters since all distribution $\mathcal{D}_k$ 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 `is.shape.ind = TRUE`. Since the Normal distribution does not have any shape parameters, lets create a two-states MSGARCH model with two single-regime GARCH processes following a unique Student-t distribution.

```
R> data("sp500ret")
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 = 50*length(spec$theta0), itermax = 500,
                  enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> fit.mle$theta

        alpha0_1   alpha1_1    beta_1   alpha0_2  alpha1_2
[1,] 0.002064724 0.04749365 0.9391432 0.05701089 0.1606847
```

```
     beta_2        nu         P         P
   0.834704 10.77268 0.2303223 0.9998429
```

```
R> transmat.mle = MSGARCH::transmat(fit.mle)
R> transmat.mle
```

```
          [,1]         [,2]
[1,] 0.2303223 0.9998429288
[2,] 0.7696777 0.0001570712
```

As we can see, the `do.shape.ind = TRUE` results in only one parameter label `nu` with no regime indication instead of having two parameters label `nu_1` and `nu_2`.

5. Distributions

We present here the conditional distribution and their functionality 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, is.its)
pdf(object, x, theta, y, is.log, is.its)
```

First of all, the `object` argument accepts a specification created with `create.spec` (see Section 2) or a fit object (see Section 6). When a specification is passed to the method, `theta` and `y` must be provided. This is not the case when a fit object is passed since the fitted object already have the relevant `theta` (which is found during optimization) and `y` (which is used for the optimization). The `cdf` and `pdf` function can be used two different ways via the `is.its` argument. When `is.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 6). When `is.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 $H_{T+1} \equiv [h_{T+1,1},h_{T+1,2},...,h_{T+1,K}]$.

5.1. The Normal distribution

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

$$f_{\mathcal{N}(0,1)}(z) \equiv \frac{1}{\sqrt{2\pi}}\ \mathrm{e}^{-\frac{1}{2}z^2}\ , \tag{8}$$

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"`.

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

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: sGARCH_normal_sym"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 0"
[1] "Default parameters:"
     alpha0 alpha1 beta
[1,]    0.1    0.1  0.8
```

5.2. The Student-t distribution

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

$$f_{\mathcal{S}(0,1,\nu)}(z) \equiv \sqrt{\frac{\nu}{\nu-2}}\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{z^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \tag{9}$$

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"`.

```
R> create.spec(model = "sGARCH", distribution = "std")
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: sGARCH_student_sym"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 nu_1
[1,]      0.1      0.1    0.8   10
```

5.3. The GED distribution

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

$$f_{\mathcal{GED}(0,1,\nu)}(z) \equiv \frac{\nu e^{-\frac{1}{2}|z/\lambda|^{\nu}}}{\lambda 2^{(1+1/\nu)}\Gamma(1/\nu)}, \tag{10}$$

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:

$$F_{\mathcal{GED}(0,1,\nu)}(z) \equiv \begin{cases} \frac{1}{2} - \frac{1}{2}F_{\mathcal{GAM}(1,1/\nu)}\left(\frac{1}{2}\left(\frac{|z|}{\lambda}\right)^{\nu}\right), & z \leq 0, \\ \\ \frac{1}{2} + \frac{1}{2}F_{\mathcal{GAM}(1,1/\nu)}\left(\frac{1}{2}\left(\frac{z}{\lambda}\right)^{\nu}\right), & z \geq 0, \end{cases} \tag{11}$$

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"`.

```
R> create.spec(model = "sGARCH", distribution = "ged")
```

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: sGARCH_ged_sym"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 nu
[1,]      0.1      0.1    0.8  2
```

5.4. Skewed distributions

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

$$f_{\mathcal{D}skew(\xi,\lambda)} \equiv \frac{2\sigma_\xi^2}{\xi + \xi^{-1}} f_{\mathcal{D}sym(\lambda)}(z_\xi), \quad z_\xi \equiv \begin{cases} \xi^{-1}(\sigma_\xi z + \mu_\xi) & \text{if} \quad z \geq -\mu_\xi/\sigma_\xi \\ \xi(\sigma_\xi z + \mu_\xi) & \text{if} \quad z < -\mu_\xi/\sigma_\xi \,, \end{cases} \tag{12}$$

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`.

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

```
[1] "Specification Type: Single-Regime"
[1] "Specification Name: sGARCH_normal_skew"
[1] "Number of parameters in variance model: 3"
[1] "Number of parameters in distribution: 1"
[1] "Default parameters:"
     alpha0_1 alpha1_1 beta_1 xi
[1,]      0.1      0.1    0.8  1
```

6. Fitting

Fitting 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 $\text{prior}(\Theta) + \text{prior}(\Lambda) + \text{prior}(\mathbf{P}) + L(y|\Theta, \Lambda, \mathbf{P})$ where L is the likelihood of y given the parameter Θ, Λ, and $\mathbf{P}$. The prior is different for each conditional variance process (see Section 3), conditional distribution (see Section 5), and type of model (see Section 4). 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 1. If any of these conditions is not respected, the prior return `-1e10` which means that the optimizer or sampler will know that the parameter draw is not a good candidate.

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

$$LLH(\theta, \lambda \,|\, \mathbf{y}) \equiv \sum_{t=1}^{T} \ln \left(f_t(y_t \,|\, \theta, \lambda) \right) \,, \tag{13}$$

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):

$$LLH(\Theta, \Lambda, \mathbf{P}|\mathbf{y}) \equiv \sum_{t=1}^{T} \ln \left(\psi_t' \mathbf{f}_t(y_t|\Theta, \Lambda) \right) \,, \tag{14}$$

where the vector ψ_t contains the conditional 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 ψ_t is easily compute with the Hamilton filter:

$$\begin{aligned} \mathring{\psi}_t \equiv& \frac{\psi_{t-1} \circ \mathbf{f}_t(y_t|\Theta, \Lambda)}{\boldsymbol{\iota}'(\psi_{t-1} \circ \mathbf{f}_t(y_t|\Theta, \Lambda))} \\ \psi_t \equiv& \mathbf{P} \mathring{\boldsymbol{\psi}}_t \,. \end{aligned} \tag{15}$$

The R package **MSGARCH** allows Maximum likelihood and Bayesian estimation. Other custom optimization schemes than those presented below can be used with the `kernel` function.

6.1. 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("sp500ret")
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 = 50*length(spec$theta0), itermax = 500, enhance.thet
R> set.seed(123)
R> fit.mle = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = ctr.mle)
R> fit.mle$theta
```

```
        alpha0_1   alpha1_1    beta_1   alpha0_2  alpha1_2
[1,] 0.001598108 0.02710623 0.890987 0.03791289 0.1178041
        beta_2            P         P
     0.8776865 9.720447e-12 0.3684928
```

```
R> transmat.mle = MSGARCH::transmat(fit.mle)
R> transmat.mle
```

```
             [,1]      [,2]
[1,] 9.720447e-12 0.3684928
[2,] 1.000000e+00 0.6315072
```

```
R> kernel(fit)
```

```
[1] 17561.94
```

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` indicate if there is a pre-optimization with the R package **DEoptim**. For simpler specification such as single-regime specification, setting `do.init = FALSE` and skipping the initialization step should provide a good estimator. The argument `NP` sets the number of vector of parameters in the population while `itermax` sets maximum iteration (number of population generation). Please refers to the **DEoptim** documentaion for more details. Finally, `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 of the rolling windows of `y`.

6.2. Bayesian estimation

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

```
R> data("sp500ret")
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, enhance.theta0 = TRUE)
R> fit.bay= MSGARCH::fit.bayes(spec = spec, y = sp500, ctr = ctr.bay)
R> tail(fit.bay$theta, 5)
```

```
Markov Chain Monte Carlo (MCMC) output:
Start = 995
End = 1000
Thinning interval = 1
        alpha0_1   alpha1_1    beta_1   alpha0_2  alpha1_2    beta_2         P         P
```

```
[1,] 0.003439296 0.02853460 0.8969148 0.05992677 0.1589380
     0.8355912 0.08925331 0.4658495
[2,] 0.004302590 0.03086301 0.8913955 0.03860429 0.1013273
     0.8951919 0.09835719 0.4902393
[3,] 0.003788251 0.03944870 0.8738322 0.04750282 0.1271350
     0.8687556 0.05205721 0.4934943
[4,] 0.005474999 0.04021973 0.8644060 0.04393671 0.1193984
     0.8759162 0.02651751 0.4994400
[5,] 0.005681199 0.03564410 0.8739715 0.05013316 0.1275241
     0.8677106 0.00996118 0.4867897
[6,] 0.005036135 0.03799062 0.8748173 0.04435196 0.1201968
     0.8763344 0.05618710 0.4997226
```

```
R> kern = kernel(fit)
R> tail(kern, 5)
```

```
[1] -6512.832 -6509.388 -6509.637 -6509.639 -6509.924
```

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 `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** chain meaning that all function for chain analysis available in R package **coda** are available.

7. 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 accept a specification created with `create.spec` (see Section 2) or a fit object (see Section 6). When a specification is passed to the method, `theta` and `y` must be provided which is not the case when a fit object is passed to object. The function `ht` outputs the filtered volatility of each regime by simply running the variance update function (See Section 3) of each single-regime specification up to time $T + 1$. The function `Pstate` runs the Hamilton filter (See Section 6) and gives the states probabilities up to time $T + 1$. These functions are useful since we can see the evolution of the volatilities and state probabilities in time.

8. 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 of the vector of observation `y`. The argument `n` sets the length of the individual simulation while the argument `m` sets the number of simulation. If a matrix of parameters such as a MCMC chain (see Section 6.2) is passed to `theta`, we automatically sets `m = M` where `M` is the number rows in the chain. Finally, the argument `burnin` unique to `sim` sets the number of discarded simulation since the state probabilities starts at $1/K$ where K is the number of states. This could be useful since the long-term probabilities of being in a given state is usually not $1/K$.

9. Predictive density

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, 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. When `is.its = TRUE`, we calculate:

$$f(\mathbf{y}) \equiv \frac{\sum_{n=1}^{N} f(\mathbf{y}|\mathbf{H_n}, \boldsymbol{\psi_n}, \Lambda_n)}{N} \tag{16}$$

where N is the total number of rows in the MCMC chain. When `is.its = FALSE`, we calculate:

$$f(x) \equiv \frac{\sum_{n=1}^{N} f(\mathbf{x}|H_{T+1,n}, \psi_{T+1,n}, \Lambda_n)}{N} \tag{17}$$

When a single vector of parameters is passed to `theta` the results are the same as calling the function `pdf` (see Section(5)).

10. Probability integral transform

The probability integral transform is sometime useful when we want to do test such as the Berkowitz test (Berkowitz 2001).

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

The difference between the `cdf` and the `pit` function is that when a matrix of parameters is passed to `theta`, each parameter are not evaluated individually. That is, we average the resulting CDF of each cumulative estimation. When `is.its = TRUE`, we calculate:

$$F(\mathbf{y}) \equiv \frac{\sum_{n=1}^{N} F(\mathbf{y}|\mathbf{H_n}, \boldsymbol{\psi_n}, \Lambda_n)}{N} \tag{18}$$

where N is the total number of rows in the MCMC chain. When `is.its = FALSE`, we calculate:

$$F(x) \equiv \frac{\sum_{n=1}^{N} F(\mathbf{x}|H_{T+1,n}, \psi_{T+1,n}, \Lambda_n)}{N} \tag{19}$$

The `do.norm` argument serves as to transform the PIT into standard Normal variate. When a single vector of parameters is passed to `theta` the results are the same as calling the function `cdf` (see Section(5)).

11. 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 estimators in-sample and at $T + 1$.

```
risk(object, theta, y, level, ES, is.its)
```

The `object` argument works as in the previous function. The argument `level` is a vector that sets at which levels the risk estimators are evaluated. The argument `ES` indicates if the ES is computed. Finally, the argument `is.its` indicates if in-sample evaluation or $T+1$ evaluation is done.

The function can calculate the risk estimators for Maximum likelihood and Bayesian estimation. We first extract the predictive density $f(x)$ given the information at time t. This is essentially the `pred` function with `is.its = FALSE`. When `is.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 `is.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 level α:

$$0 \equiv \int_{-\infty}^{VaR_\alpha} f(x)dx - \alpha \tag{20}$$

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 integrating:

$$ES_\alpha \equiv \int_{-\infty}^{VaR_\alpha} xf(x)dx\,. \tag{21}$$

We use the R function `integrate` from the `stats` namespace to do so.

12. Information criterion

12.1. AIC

The Akaike information criterion (AIC) Akaike (1974) is a measure of the relative quality of statistical models for a given set of data.

AIC(fit)

The AIC is written as:

$$AIC \equiv 2k - 2LLH(\mathbf{y}|\Theta, \Lambda, \mathbf{P})\,. \tag{22}$$

Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. AIC rewards goodness of fit (as assessed by the likelihood function), but it also includes a penalty that is an increasing function of the number of estimated parameters. This prevents overfitting. When a Bayesian fit is passed to the `AIC` function, the the AIC on the posterior mean $\bar{\Theta}$, $\bar{\Lambda}$, and $\bar{\mathbf{P}}$ is calculated.

12.2. BIC

The Bayesian information criterion (BIC) (Schwarz *et al.* 1978) is a criterion for model selection among a finite set of models. The model with the lowest BIC is preferred.

BIC(fit)

The BIC is written as:

$$BIC \equiv k\ln(n) - 2LLH(\mathbf{y}|\Theta, \Lambda, \mathbf{P})\,. \tag{23}$$

Both BIC and AIC prevents overfitting by introducing a penalty term to the number of parameters 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` function, the the BIC on the posterior mean $\bar{\Theta}$,$\bar{\Lambda}$, and $\bar{\mathbf{P}}$ is calculated.

12.3. DIC

The deviance information criterion (Gelman *et al.* 2014) is particularly useful in Bayesian model selection problems where the posterior distributions of the models have been obtained by MCMC simulation (see Section 6).

DIC(fit)

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 $p_D \equiv \frac{1}{2}\widehat{\text{var}}\,(D(\Theta, \Lambda, \mathbf{P}))$. 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. The deviance information criterion is calculated as:

$$DIC \equiv p_D + \bar{D} \tag{24}$$

13. Empirical illustration

We consider the daily log returns of Amazon inc. from 1998-01-01 to 2015-12-31. The data are retreived from Yahoo finance. The daily log return are calculated as $y_t \equiv ()\ln P_t - \ln P_{t-1}) * 100$, where P_t is the price index at time t. We analyze the in-sample fitting of a single-regime GJR with a skewed Student-t conditional distribution and a two states Markov-switching with two GJR processes with both regimes having a skewed Student-t conditional distribution.

[Insert Figure 1 about here.]

13.1. MLE

We first fit both models with the pre-optimization argument `do.init = TRUE` and the argument `enhance.theta0 = TRUE`.

```
R> require(MSGARCH)
R> data("AMZN")
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 = 50*length(spec1$theta0),
                   itermax = 500, enhance.theta0 = TRUE)
R> set.seed(123)
R> fit.mle1 = MSGARCH::fit.mle(spec = spec1, y = AMZN, ctr = ctr.mle1)
R> fit.mle1$theta
```

```
       alpha0_1   alpha1_1    alpha2_1    beta_1     nu_1     xi_1
[1,] 0.01127067 0.01270524 0.006689363 0.9835743 3.400922 1.027414
```

```
R> MSGARCH::unc.vol(fit.mle1)
```

```
[1] 4.833484
```

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

```
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, enhance.theta0 = TRUE)
```

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

```
       alpha0_1     alpha1_1  alpha2_1    beta_1     nu_1
[1,] 0.08881201 9.123837e-06 0.2122064 0.8934922 14.37109
         xi_1    alpha0_2    alpha1_2     alpha2_2    beta_2
     1.124587 0.007843311 0.008476387 0.0001073781 0.9913015
         nu_2     xi_2         P         P
     2.986982 1.005895 0.9515958 0.0277085
```

```
R> MSGARCH::unc.vol(fit.mle2)
```

```
        [,1]     [,2]
[1,] 3.46768 6.814331
```

```
R> state = MSGARCH::Pstate(fit.mle2)
R> plot(state)
R> ht = MSGARCH::ht(fit.mle2)
R> plot(ht)
```

[Insert Figure 2, Figure 3, Figure 4 about here.]

We note the single-regime model induces a high level of volatility persistence (see Figure 2). Both regime of the MSGARCH specification are also very persistent (see Figure 4), but the regime switching framework makes the overall process less persistent. We can also observe that the `alpha2_1` parameter is very high in the first regime showing a considerable leverage effect. 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.

13.2. Bayesian estimation

For completeness, we also do Bayesian estimation using as starting value the previously found Maximum likelihood estimators.

```
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 = AMZN, ctr = ctr.bay1)
```

```
R> tail(fit.bay1$theta, 5)
```

```
Markov Chain Monte Carlo (MCMC) output:
Start = 995
End = 1000
Thinning interval = 1
        alpha0_1    alpha1_1    alpha2_1    beta_1    nu_1     xi_1
[1,] 0.018999604 0.01452825 0.006612186 0.9810190 3.398064 1.032599
[2,] 0.018893202 0.01070068 0.021588991 0.9779775 3.421657 1.016574
[3,] 0.012589833 0.01119870 0.016017176 0.9801415 3.415150 1.019992
[4,] 0.009050485 0.01278142 0.010105783 0.9818406 3.406652 1.024718
[5,] 0.011441447 0.01082906 0.010848093 0.9834050 3.408737 1.024203
[6,] 0.013330807 0.01314357 0.007811320 0.9822665 3.402145 1.028435

R> uncvol =  MSGARCH::unc.vol(fit.bay1)
R> tail(uncvol, 5)

[1] 5.069430 3.877797 4.386939 4.810499 4.048494

R> ht = MSGARCH::ht(fit.bay1)
R> plot(ht)

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 = AMZN, ctr = ctr.bay2)

R> tail(fit.bay2$theta, 5)

Markov Chain Monte Carlo (MCMC) output:
Start = 995
End = 1000
Thinning interval = 1
       alpha0_1    alpha1_1  alpha2_1      beta_1       nu_1
         xi_1     alpha0_2     alpha1_2   alpha2_2     beta_2
         nu_2        xi_2           P           P
[1,] 0.07871532 0.003297127 0.2242508   0.8879337   14.38865
     1.114410   0.02327884  0.009407066 0.009033077 0.9854262
     2.967730   1.0205691   0.9539879   0.02889543
[2,] 0.06092543 0.002829539 0.2110616   0.8948067   14.40955
     1.105554   0.01692709  0.008996187 0.009200839 0.9859449
     2.989937   1.0026074   0.9636504   0.02223763
[3,] 0.07146154 0.003801530 0.2200072   0.8884499   14.40702
     1.108791   0.01537451  0.006832919 0.010279880 0.9876950
     2.983265   0.9992740   0.9567146   0.01420153
[4,] 0.06575896 0.001979360 0.2155976   0.8886385   14.41247
     1.105823   0.01635933  0.007136613 0.009448035 0.9876330
     2.987376   0.9986458   0.9585287   0.01552453
```

```
[5,] 0.06515876 0.003026139 0.2134307   0.8907645   14.40740
     1.107170   0.01824112  0.007493229 0.008644758 0.9876047
     2.985081   1.0042042   0.9609832   0.01847942
[6,] 0.06865786 0.003771401 0.2146785   0.8907243   14.40030
     1.109477   0.02159964  0.006561914 0.007966619 0.9885715
     2.979423   1.0095322   0.9600202   0.01916346
```

```
R> uncvol =  MSGARCH::unc.vol(fit.bay2)
R> tail(uncvol, 5)
```

```
           [,1]     [,2]
[496,] 4.704725 5.969187
[497,] 4.175545 6.857148
[498,] 2.932871 5.734633
[499,] 3.420612 5.491580
[500,] 3.948667 4.804021
```

```
R> state = MSGARCH::Pstate(fit.bay2)
R> plot(state)
R> ht = MSGARCH::ht(fit.bay2)
R> plot(ht)
```

[Insert Figure 5, Figure 6, Figure 7 about here.]

In general we observe that all the results are similar to the Maximum likelihood estimation, but now with a range which can clearly be seen in all the figures (Figure 5, 6, and 7). We also observe that the Bayesian unconditional volatility of the MSGARCH specification tend to vary in pair, that is when the unconditional volatility of the first regime is higher than the MLE unconditional volatility of the first regime, the Bayesian uncondtional volatility of the second regime tend to be lower than the MLE unconditional volatility of the second regime. We can also observe that the unconditional volatility of the single-regime model tend to vary a lot.

13.3. Information criterion

To find the prefered model, we compute the AIC and the BIC for the Maximum likelihood esimation and the DIC for the Bayesian esimation.

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

```
[1] 22469.66 22367.21
```

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

```
[1] 22508.17 22457.07
```

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

```
[1] 22434.66 22269.62
```

We can observe that the MSGARCH specification is better in all information criteria. We can also observe that the difference in value in the DIC is quite large which could mean that Bayesian optimization did a better job at finding optimal parameters of the MSGARCH specification.

13.4. PIT

We run the pit function on all fit to see if each model fits well the data.

```
R> plot(MSGARCH::pit(fit.mle1, do.norm = FALSE, is.its = TRUE))
R> plot(MSGARCH::pit(fit.mle2, do.norm = FALSE, is.its = TRUE))
R> plot(MSGARCH::pit(fit.bay1, do.norm = FALSE, is.its = TRUE))
R> plot(MSGARCH::pit(fit.bay2, do.norm = FALSE, is.its = TRUE))

R> pit.mle1 = MSGARCH::pit(fit.mle1, do.norm = TRUE, is.its = TRUE)
R> pit.mle2 = MSGARCH::pit(fit.mle2, do.norm = TRUE, is.its = TRUE)
R> pit.bay1 = MSGARCH::pit(fit.bay1, do.norm = TRUE, is.its = TRUE)
R> pit.bay2 = MSGARCH::pit(fit.bay2, do.norm = TRUE, is.its = TRUE)

R> l.pit = length(pit.mle1$pit)
R> Decision = NULL
R> Decision[1] = rugarch::BerkowitzTest(data = pit.mle1$pit[2:l.pit],
                                         significance = 0.05)$Decision
R> Decision[2] = rugarch::BerkowitzTest(data = pit.mle2$pit[2:l.pit],
                                         significance = 0.05)$Decision
R> Decision[3] = rugarch::BerkowitzTest(data = pit.bay1$pit[2:l.pit],
                                         significance = 0.05)$Decision
R> Decision[4] = rugarch::BerkowitzTest(data = pit.bay2$pit[2:l.pit],
                                         significance = 0.05)$Decision

R> Decision
```

```
[1] "fail to reject NULL" "fail to reject NULL"
    "fail to reject NULL" "fail to reject NULL"
```

[Insert Figure 8 about here.]

The PIT can be seen in Figure 8. We make all PIT pass the Berkowitz test implemented in the function `BerkowitzTest` in the R package **rugarch**. The NULL for this test is a Normal (0,1) with no autocorrelation which all models fail to reject the NULL.

13.5. Value-at-Risk

We analyze the Value-at-Risk at 5% for all models.

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

R> tsRainbow <- rainbow(ncol(risk), alpha = 0.8)
R> colnames(risk) = c("GJR mle", "MSGARCH GJR mle",
                      "GJR bay", "MSGARCH GJR bay")
R> plot(zoo::zoo(risk),plot.type = "single",
        col = tsRainbow, ylab = "VaR",xlab = "Date")
R> legend("bottomright",legend =  colnames(risk),
           lty = 1, col = tsRainbow)
```

[Insert Figure 9 about here.]

The Value-at-Risk at 5% for both mle estimation of each model can be seen in Figure 9. They look similar except that the MSGARCH model often shows big downward spikes which can be typical for technologie stocks like Amazon inc. that are very sensible to overreaction.

14. Conclusion

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> ")`.

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.

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

Berkowitz J (2001). "Testing Density Forecasts, with Applications to Risk Management." *Journal of Business and Economic Statistics*, **19**(4), 465–474.

Black F (1976). "Studies of Stock Price Volatility Changes."

Bluteau K, Ardia D, Bout K, Peterson B (2016). ***MSGARCH**: Markov Switching GARCH Models and Extensions.* 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.

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

Christie AA (1982). "The Stochastic Behavior of Common Stock Variances: Value, Leverage and Interest Rate Effects." *Journal of financial Economics*, **10**(4), 407–432.

Creal D, Koopman SJ, Lucas A (2013). "Generalized autoregressive score models with applications." *Journal of Applied Econometrics*, **28**(5), 777–795.

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.

Francq C, Zakoian JM (2011). *GARCH models: structure, statistical inference and financial applications*. John Wiley & Sons.

Gelman A, Carlin JB, Stern HS, Rubin DB (2014). *Bayesian data analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA.

Ghalanos A (2015). ***rugarch**: Univariate GARCH Models.* R package version 1.3-6, URL https://cran.r-project.org/package=rugarch.

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.

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

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

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

Hillebrand E (2005). "Neglecting Parameter Changes in GARCH Models." *Journal of Econometrics*, **129**(1), 121–138.

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.

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.

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.

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

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.

Trottier DA, Ardia D (2016). "Moments of Standardized Fernandez-Steel Skewed Distributions: Applications to the Estimation of GARCH-Type Models." *Finance Research Letters*.

Vihola M (2012). "Robust Adaptive Metropolis Algorithm with Coerced Acceptance Rate." *Statistics and Computing*, **22**(5), 997–1008.

Zakoian JM (1994). "Threshold Heteroskedastic Models." *Journal of Economic Dynamics and Control*, **18**(5), 931–955.

Figure 1: Log-returns of Amazon inc. from 1998-01-01 to 2015-12-13.

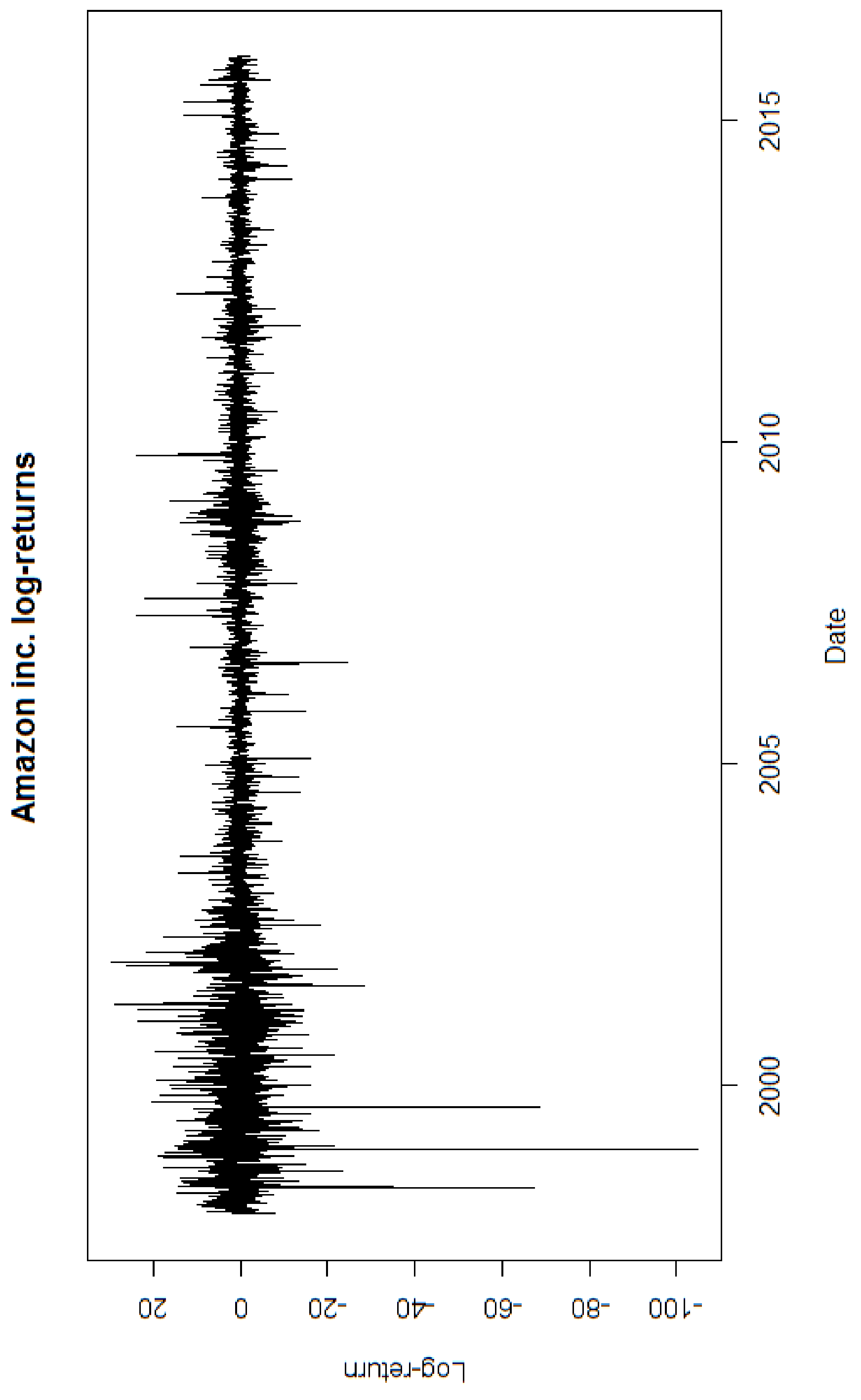


Figure 2: Conditional volatility of Amazon inc. from 1998-01-01 to 2015-12-13 from the Maximum likelihood estimation of a single-regime GJR with skewed Student-t innovations.

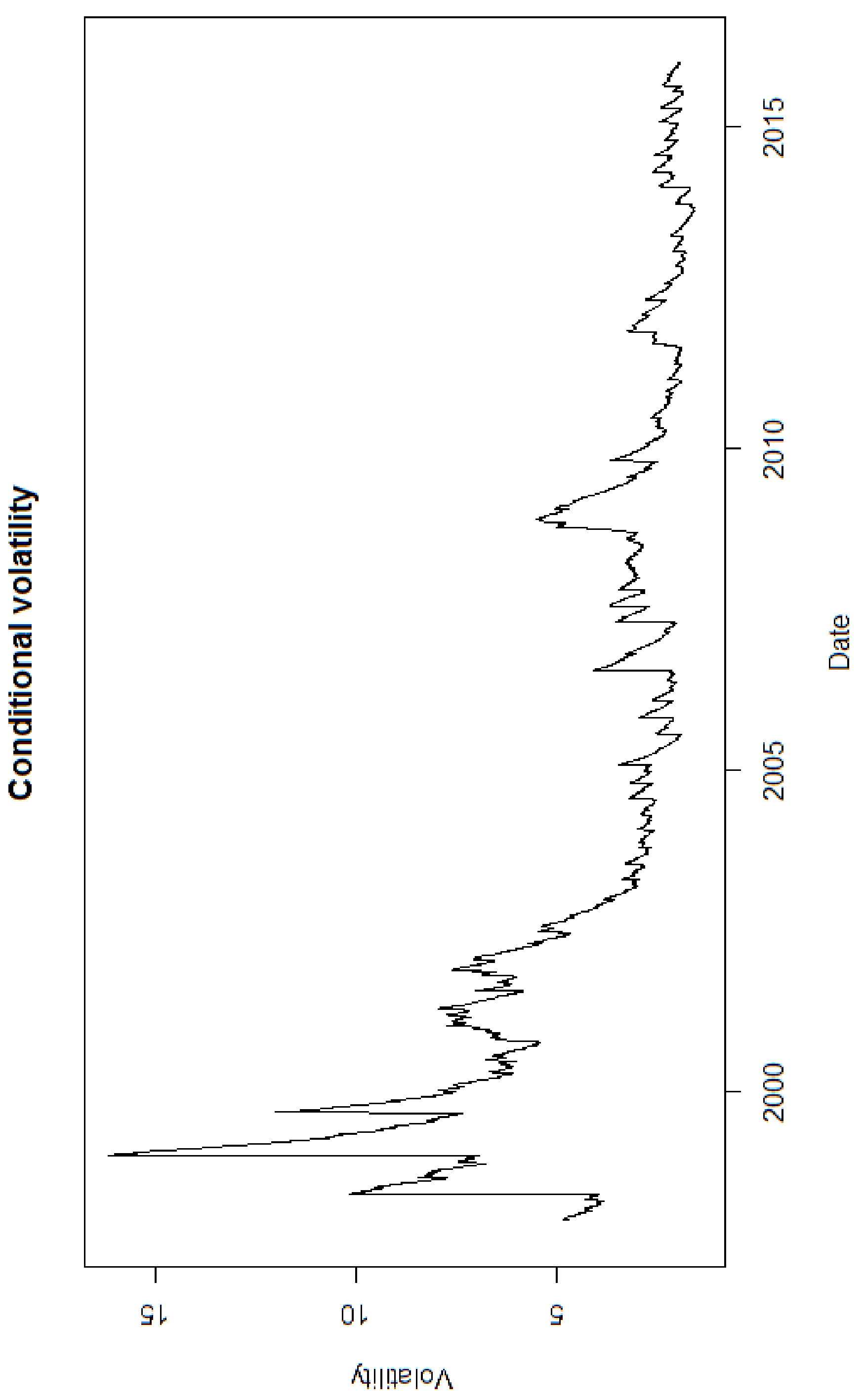


Figure 3: State probability to be in the second regime of Amazon inc. from 1998-01-01 to 2015-12-13 from the Maximum likelihood estimation of a two-states Markov-switching GJR with skewed Student-t innovations.

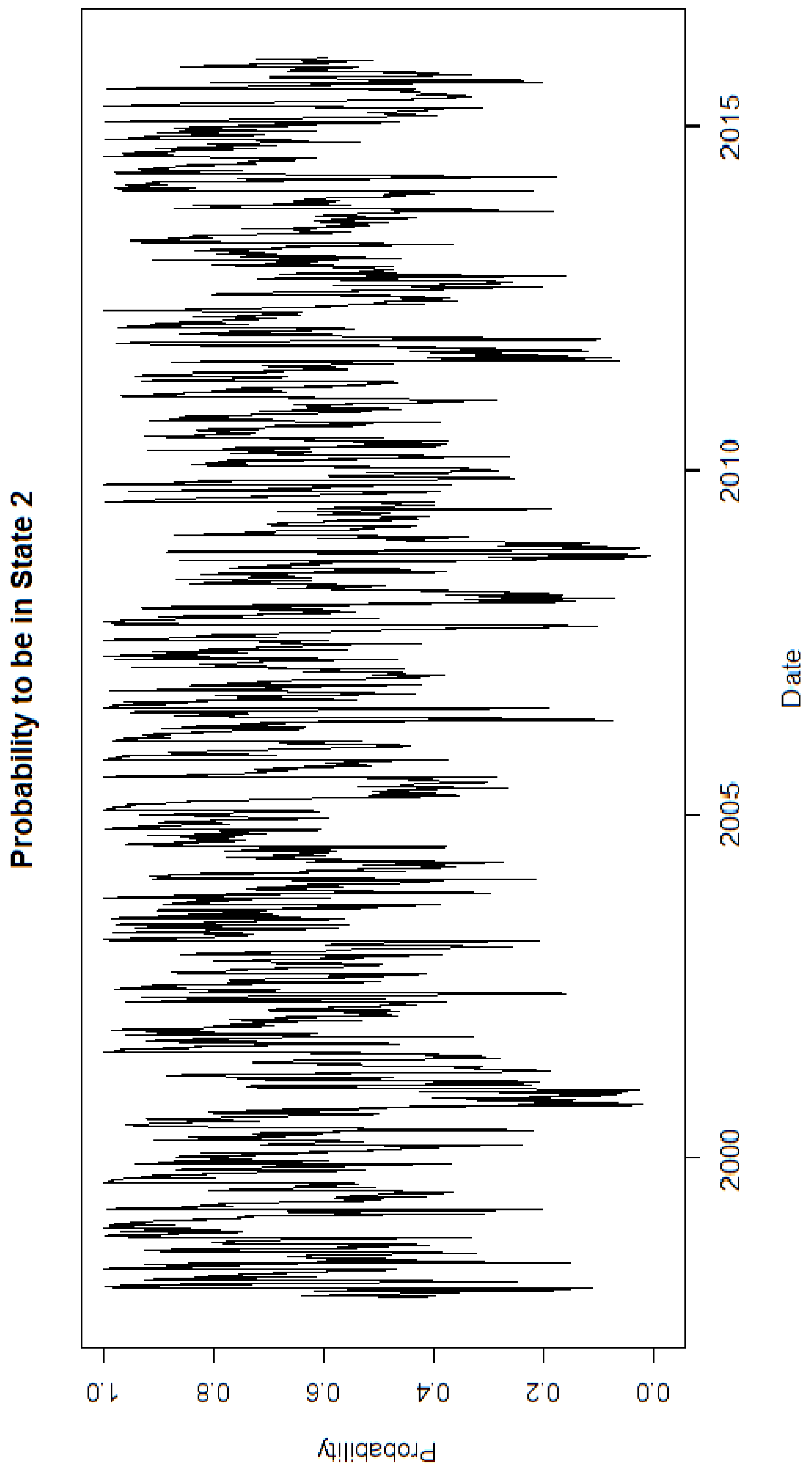


Figure 4: Conditional volatility of Amazon inc. from 1998-01-01 to 2015-12-13 from the Maximum likelihood estimation of a two-states Markov-switching GJR with skewed Student-t innovations.

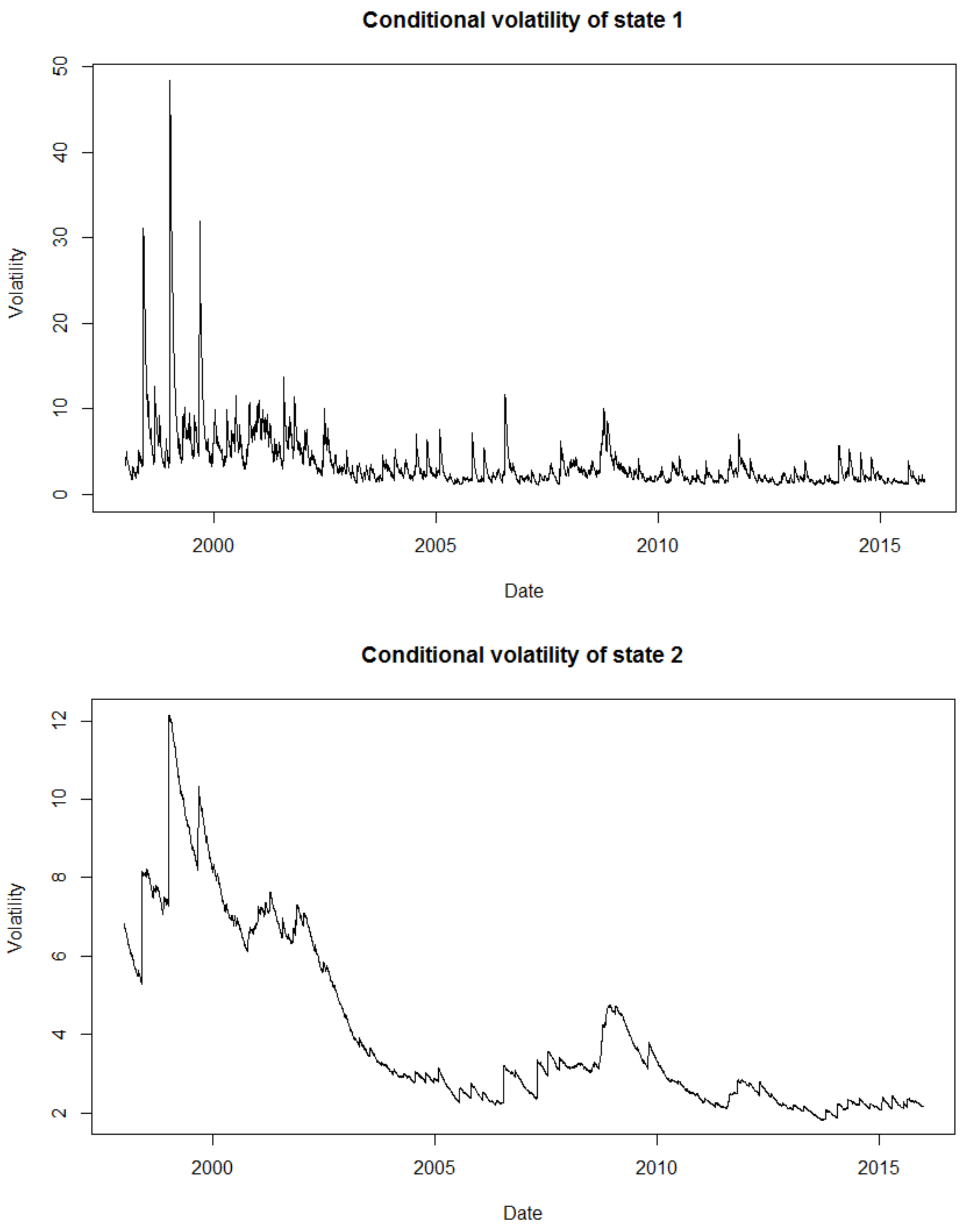


Figure 5: Conditional volatility of Amazon inc. from 1998-01-01 to 2015-12-13 from the Bayesian estimation of a single-regime GJR with skewed Student-t innovations.

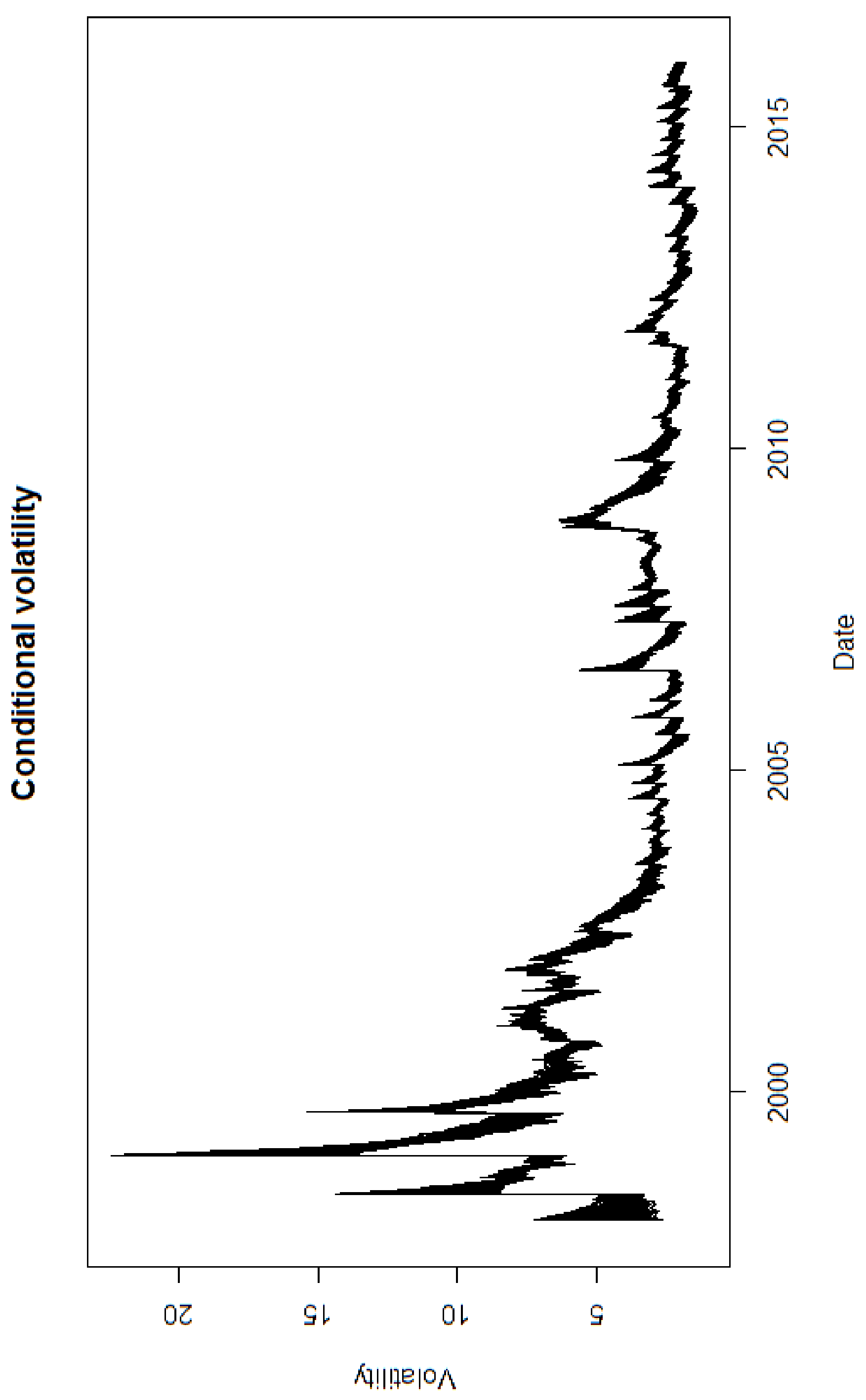


Figure 6: State probability to be in the second regime of Amazon inc. from 1998-01-01 to 2015-12-13 from the Bayesian estimation of a two-states Markov-switching GJR with skewed Student-t innovations.

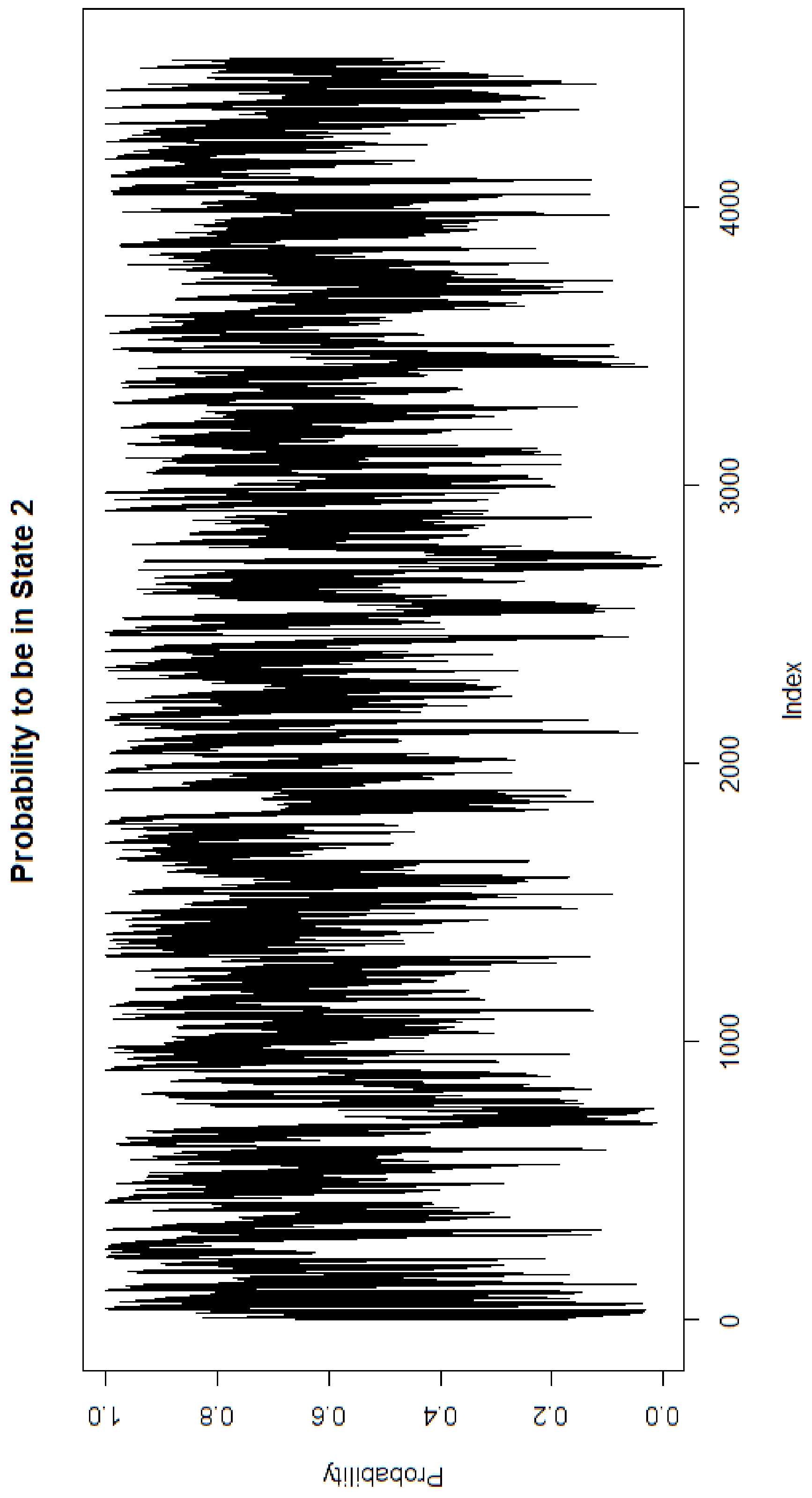
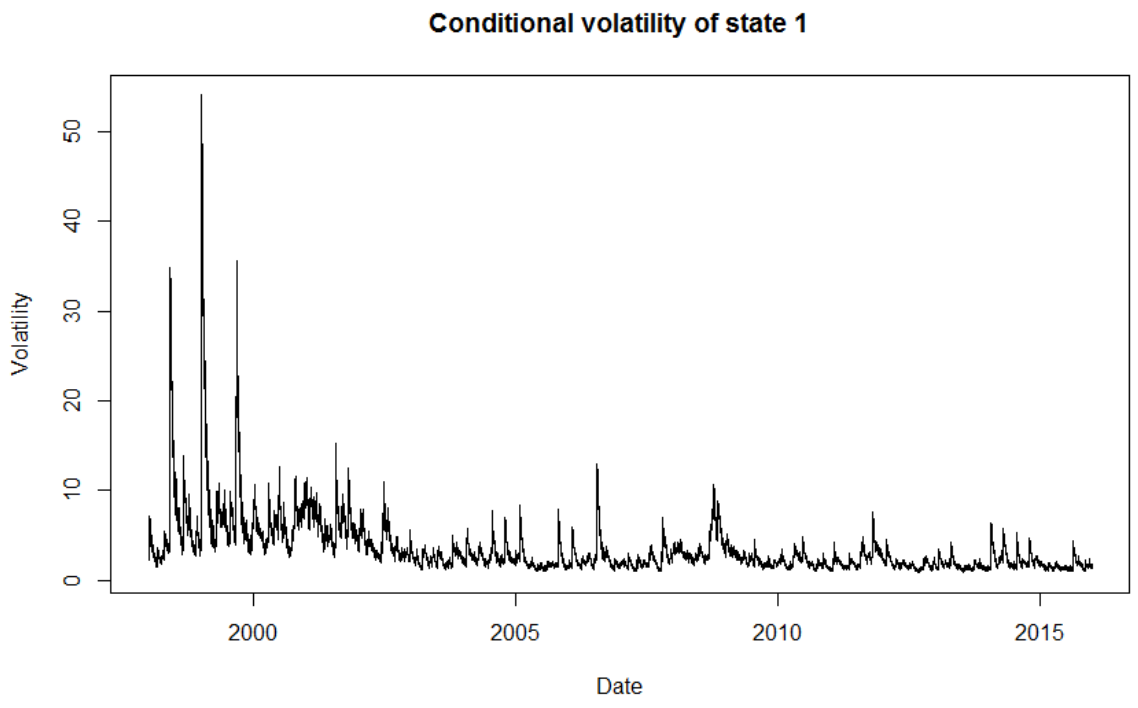


Figure 7: Conditional volatility of Amazon inc. from 1998-01-01 to 2015-12-13 from the Bayesian estimation of a two-states Markov-switching GJR with skewed Student-t innovatiosn.



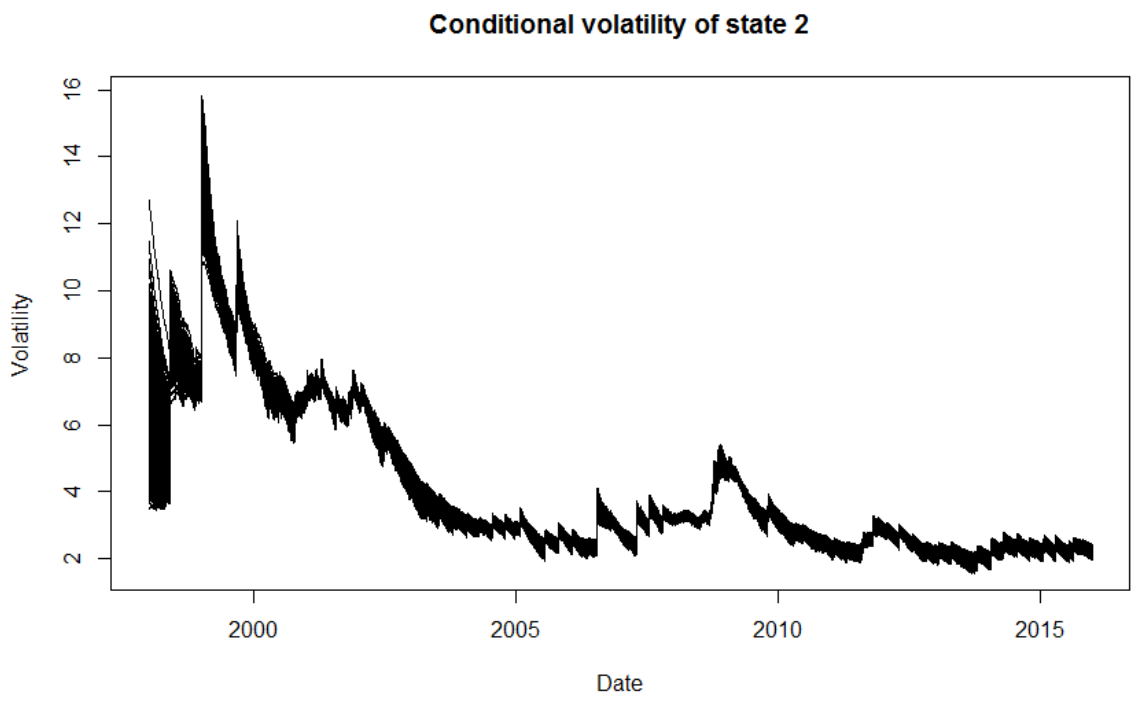


Figure 8: Probability integral transform: Upper left: Single-regime with Maximum likelihood estimation, Lower left: Single-regime with Bayesian estimation, Upper right: Markov-switching with Maximum likelihood estimation, Lower right: Markov-switching with Bayesian estimation.

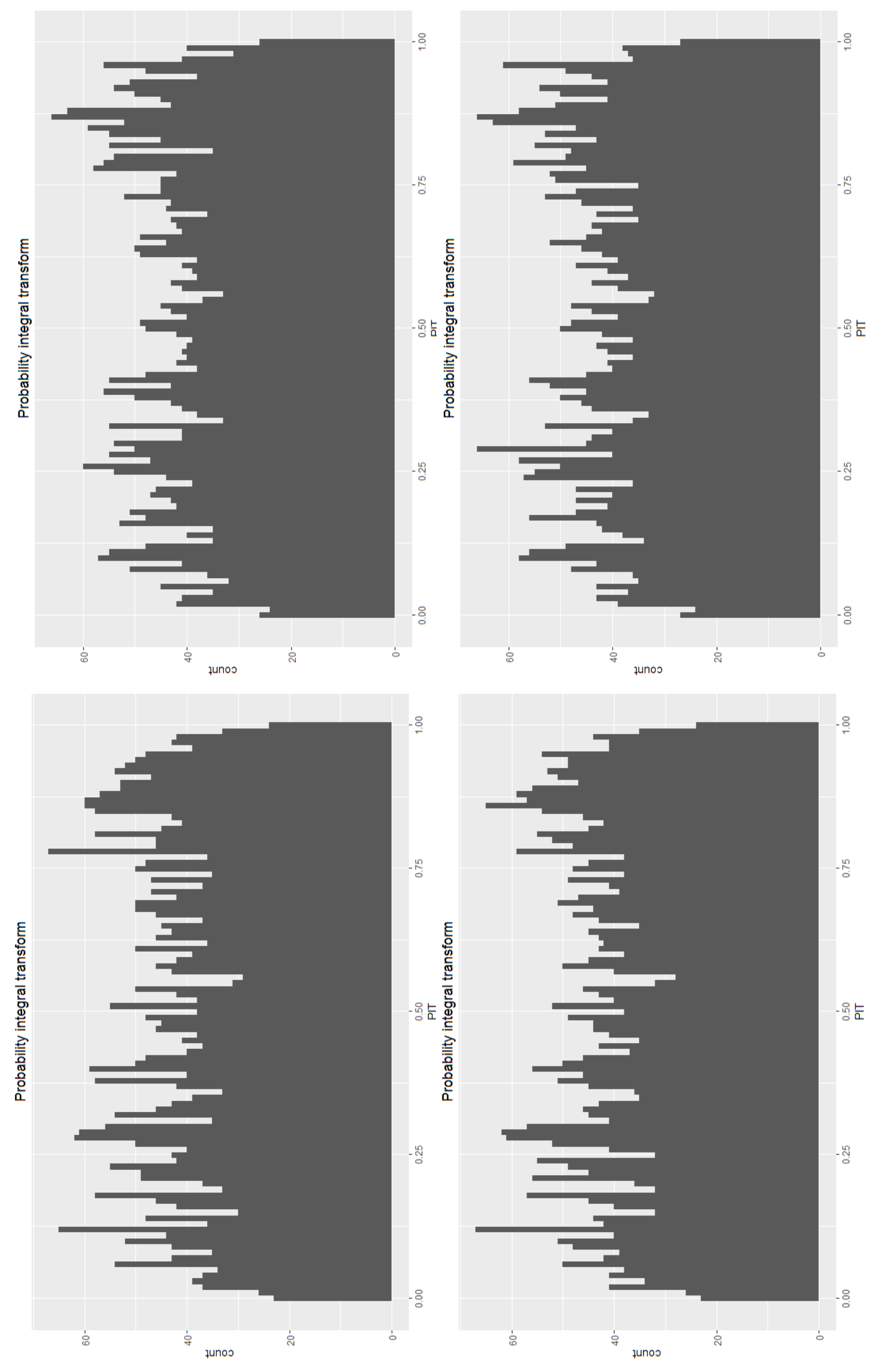
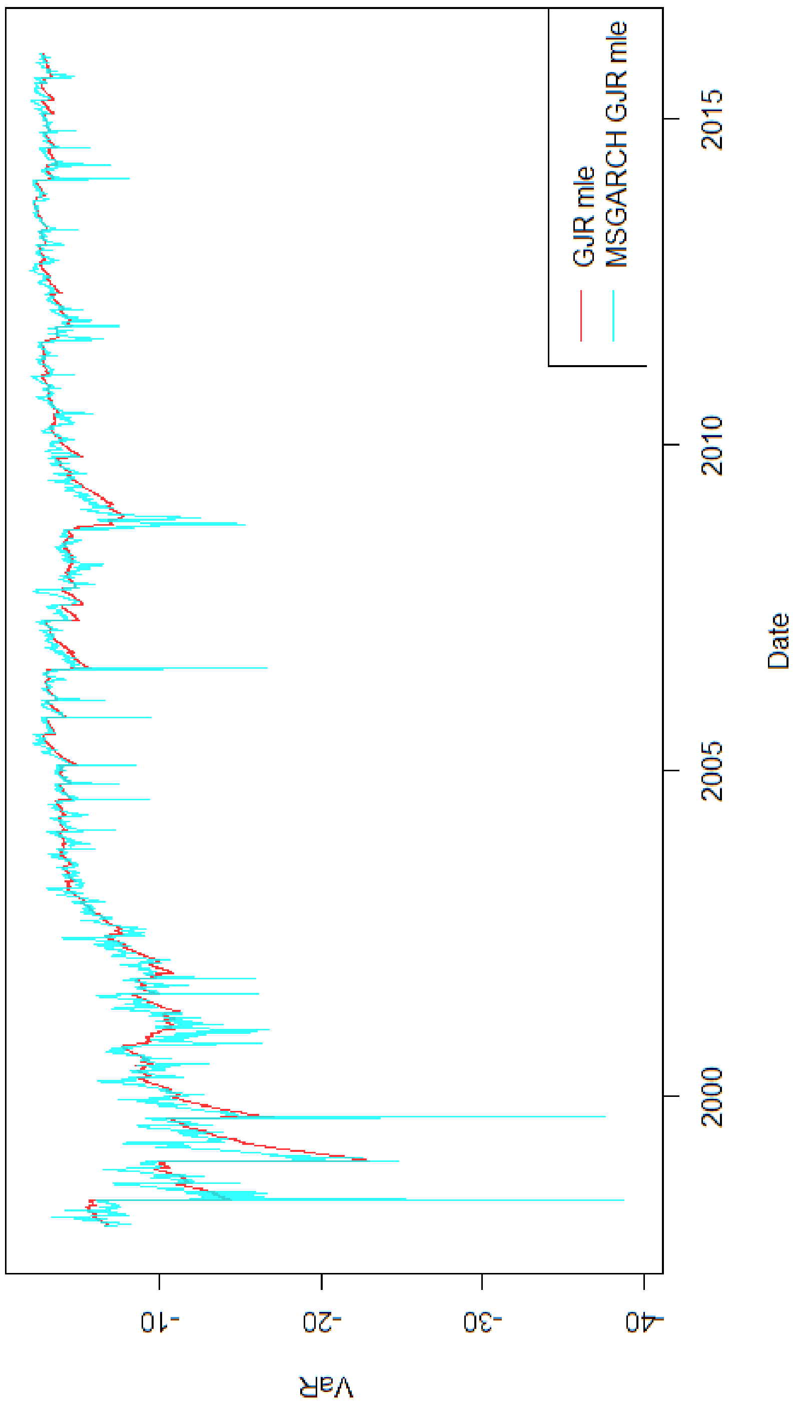


Figure 9: In-sample 5% Value-at-Risk of the fitted models



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