Journal of Statistical Software

October 2019, Volume 91, Issue 4. doi: 10.18637/jss.v091.i04

Markov-Switching GARCH Models in R: The MSGARCH Package

David Ardia
HEC Montréal

Keven Bluteau
University of Neuchâtel
Vrije Universiteit Brussel

Kris Boudt
Ghent University
Vrije Universiteit Brussel
Vrije Universiteit Amsterdam

Leopoldo Catania
Aarhus University
CREATES

Denis-Alexandre Trottier
Laval University

Abstract

We describe the package **MSGARCH**, which implements Markov-switching GARCH (generalized autoregressive conditional heteroscedasticity) models in R with efficient C++ object-oriented programming. Markov-switching GARCH models have become popular methods to account for regime changes in the conditional variance dynamics of time series. The package **MSGARCH** allows the user to perform simulations as well as maximum likelihood and Bayesian Markov chain Monte Carlo estimations of a very large class of Markov-switching GARCH-type models. The package also provides methods to make single-step and multi-step ahead forecasts of the complete conditional density of the variable of interest. Risk management tools to estimate conditional volatility, value-at-risk, and expected-shortfall are also available. We illustrate the broad functionality of the **MSGARCH** package using exchange rate and stock market return data.

Keywords: GARCH, MSGARCH, Markov-switching, conditional volatility, forecasting, R software.

1. Introduction

In 2003, Robert Engle received the Nobel Prize for his "contribution to methods of analyzing economic time series with time-varying volatility (ARCH)" (Nobel Media 2003). The seminal paper introducing the orginal autoregressive conditional heteroscedasticity (ARCH) model is

Engle (1982), while its generalization to GARCH was introduced by Bollerslev (1986). Since then, multiple extensions of the GARCH scedastic function have been proposed to capture additional stylized facts observed in financial and economic time series, such as nonlinearities, asymmetries, and long-memory properties; see Teräsvirta (2009) for a review. According to the *Time Series Analysis* (Hyndman 2019) and *Empirical Finance* (Eddelbuettel 2019) task views at `https://CRAN.R-project.org/web/views`, the following implementations of univariate GARCH-type models are available in the R (R Core Team 2018) programming language: **bayesGARCH** (Ardia and Hoogerheide 2010), **fGarch** (Wuertz, Chalabi, Miklovic, Boudt, and Chausse 2016), **GAS** (Ardia, Boudt, and Catania 2019b), **gets** (Pretis, Reade, and Sucarrat 2018), **GEVStableGarch** (Sousa, Otiniano, Lopes, and Diethelm 2015), **lgarch** (Sucarrat 2015), **rugarch** (Ghalanos 2017) and **tseries** (Trapletti and Hornik 2017). In GARCH-type models, the conditional volatility is driven by shocks in the observed time series. An alternative approach is to assume that volatility is driven by volatility-specific shocks. This is the case in stochastic volatility models, as available in the R package **stochvol** (Kastner 2016).

Recent studies show that volatility predictions by GARCH-type models may fail to capture the true variation in volatility in the case of regime changes in the volatility dynamics (see, e.g., Lamoureux and Lastrapes 1990; Bauwens, Backer, and Dufays 2014). A solution to this problem is to allow the parameters of the GARCH model to vary over time according to a latent discrete Markov process. This approach is called the Markov-switching GARCH (MSGARCH) model, which leads to volatility forecasts that can quickly adapt to variations in the unconditional volatility level.

MSGARCH models are mainly used in finance. Other fields of applications include the analysis of business cycles in economics (see, e.g., Kim and Nelson 1999) and the forecasting of windpower (see, e.g., Zhang, Wang, and Wang 2014). Their use is not widespread in industry applications mainly because their implementation is tedious and their estimation is not trivial. We fill this gap with the R package **MSGARCH** (Ardia, Bluteau, Boudt, Catania, Ghalanos, Peterson, and Trottier 2019a), and provide researchers and practitioners with a ready-to-use program within the R environment. The R package **MSGARCH** implements the specification of Haas, Mittnik, and Paolella (2004a), which, compared to the MSGARCH model in the seminal work of Gray (1996), has the advantage that the estimation does not face the "path-dependency problem".[1] In their model, the authors define K *separate* GARCH-type processes, one for each regime of the unobserved Markov chain. In addition to its appealing computational aspects, the MSGARCH model of Haas *et al.* (2004a) has conceptual advantages. Indeed, one reason for specifying Markov-switching models that allow for different GARCH behavior in each regime is to capture the difference in the variance dynamics of low- and high-volatility periods.

The R package **MSGARCH** provides a comprehensive set of methods for estimating, simulating, and forecasting with MSGARCH models. It includes the possibility of specifying different GARCH processes and conditional distributions for each state. Evaluation of risk management metrics such as the value-at-risk and the expected-shortfall is also available. The

[1]Literature on computationally feasible implementations of MSGARCH models started with the seminal work of Gray (1996), Dueker (1997), and Klaassen (2002), who proposed different solutions to the "path-dependency problem" raised by Cai (1994) and Hamilton and Susmel (1994). It means that, for a sample of size T and a MSGARCH model with K regimes, the evaluation of K^T different volatility paths is required, rendering the estimation intractable for medium and large sample sizes.

R package **MSGARCH** relies on object-oriented programming techniques in C++ via the R packages **Rcpp** (Eddelbuettel, François, Allaire, Ushey, Kou, Bates, and Chambers 2018; Eddelbuettel and François 2011) and **RcppArmadillo** (Eddelbuettel, François, and Bates 2017; Eddelbuettel and Sanderson 2014). Efficient implementation is key, especially when backtesting models in risk management.

The package's usage and the available methods are similar to those of the R package **rugarch** (Ghalanos 2017). First, the user creates a model specification. Second, she/he estimates the model. Third, she/he can perform predictions and risk forecasts. Plotting, summary and model discrimination methods can be applied to the fitted model. The R package **MSGARCH** is available from the Comprehensive R Archive Network (CRAN) repository at `https://CRAN.R-project.org/package=MSGARCH` and the development version is available on the **MSGARCH** website at `http://keblu.github.io/MSGARCH/`.

The outline of the paper is as follows. Section 2 introduces the models. Section 3 describes the R implementation and serves as a brief user manual. Section 4 illustrates the package's usage on an application using Swiss market index data. Section 5 concludes.

2. Markov-switching GARCH models

Denote the variable of interest at time t by y_t. We assume that y_t has zero mean and is not serially correlated, that is, the following moment conditions are assumed: $\mathsf{E}[y_t] = 0$ and $\mathsf{E}[y_t y_{t-l}] = 0$ for $l \neq 0$ and all $t > 0$. This assumption is realistic for high frequency returns for which the (conditional) mean is often assumed to be zero; see, for instance, McNeil, Frey, and Embrechts (2015). In other applications, the assumption that the MSGARCH process has conditional mean zero often implies applying the MSGARCH model to a de-meaned time series.[2] In the most simple case of a constant mean, this is the series in excess of the sample mean. When the series has dynamics in the conditional mean, the de-meaned time series are the residuals of a time series model, like an ARFIMAX, as in the R package **rugarch** (Ghalanos 2017). Since the **MSGARCH** package decouples the mean and volatility estimation, GARCH-in-mean models and joint approaches to mean-variance switching as in Kim and Nelson (1999) are not possible.

We allow for regime-switching in the conditional variance process. Denote by $\mathcal{I}_{t-1}$ the information set observed up to time $t-1$, that is, $\mathcal{I}_{t-1} \equiv \{y_{t-i}, i > 0\}$. The general Markov-switching GARCH specification can then be expressed as:

$$y_t \,|\, (s_t = k, \mathcal{I}_{t-1}) \sim \mathcal{D}(0, h_{k,t}, \boldsymbol{\xi}_k)\,, \tag{1}$$

where $\mathcal{D}(0, h_{k,t}, \boldsymbol{\xi}_k)$ is a continuous distribution with zero mean, time-varying variance $h_{k,t}$, and additional shape parameters gathered in the vector $\boldsymbol{\xi}_k$.[3] The integer-valued stochastic variable s_t, defined on the discrete space $\{1, \ldots, K\}$, characterizes the Markov-switching GARCH model. We define the standardized innovations as $\eta_{k,t} \equiv y_t / h_{k,t}^{1/2} \overset{iid}{\sim} \mathcal{D}(0, 1, \boldsymbol{\xi}_k)$.

[2] Furthermore, assuming a non zero state dependent mean would generally imply $\mathsf{E}[y_t y_{t-1}] \neq 0$, which is against the model we specified.

[3] As explained below, the parametric formulation of the conditional distribution $\mathcal{D}(0, h_{k,t}, \boldsymbol{\xi}_k)$ can be different across regimes. In this case, the notation $\mathcal{D}_k(0, h_{k,t}, \boldsymbol{\xi}_k)$ would be more appropriate. The same applies for the $h(\cdot)$ function in (2). We keep the simpler notation to improve readability. Also, for $t = 1$, we initialize the regime probabilities and the conditional variances at their unconditional levels. To simplify exposition, we use henceforth for $t = 1$ the same notation as for general t, since there is no confusion possible.

2.1. State dynamics

The R package **MSGARCH** package implements two approaches to the dynamics of the state variable, namely the assumption of a first-order ergodic homogeneous Markov chain which characterizes the Markov-switching GARCH model of Haas *et al.* (2004a), and the assumption of independent draws from a multinomial distribution which characterizes the mixture of GARCH models of Haas, Mittnik, and Paolella (2004b).

First-order Markov chain

We assume that s_t evolves according to an unobserved first-order ergodic homogeneous Markov chain with $K \times K$ transition probability matrix $\mathbf{P}$:

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

where $p_{i,j} \equiv \mathsf{P}[s_t = j \,|\, s_{t-1} = i]$ is the probability of a transition from state $s_{t-1} = i$ to state $s_t = j$. Obviously, the following constraints hold: $0 < p_{i,j} < 1 \; \forall i, j \in \{1, \ldots, K\}$, and $\sum_{j=1}^{K} p_{i,j} = 1$, $\forall i \in \{1, \ldots, K\}$. Given the parametrization of $\mathcal{D}(\cdot)$, we have $\mathsf{E}[y_t^2 \,|\, s_t = k, \mathcal{I}_{t-1},] = h_{k,t}$, that is, $h_{k,t}$ is the variance of y_t conditional on the realization of $s_t = k$.

In the MSGARCH model of Haas *et al.* (2004a), the conditional variances $h_{k,t}$ for $k = 1, \ldots, K$ are assumed to follow K *separate* GARCH-type processes which evolve in parallel.

Independent states

A related specification has been also introduced by Haas *et al.* (2004b) and is referred to as the mixture of GARCH. In this case, we assume that s_t is sampled independently over time from a Multinomial distribution with support $\{1, \ldots, K\}$ and vector of probabilities $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_K)^\top$, that is $\mathsf{P}[s_t = k] = \omega_k$. We have that the same parametric formulation of (1) with K *separate* GARCH-type models defined for each component of the mixture.[4]

2.2. Conditional variance dynamics

As in Haas *et al.* (2004a), the conditional variance of y_t is assumed to follow a GARCH-type model. Hence, conditionally on regime $s_t = k$, $h_{k,t}$ is available as a function of the past observation, y_{t-1}, past variance $h_{k,t-1}$, and the additional regime-dependent vector of parameters $\boldsymbol{\theta}_k$:

$$h_{k,t} \equiv h(y_{t-1}, h_{k,t-1}, \boldsymbol{\theta}_k)\,, \tag{2}$$

where $h(\cdot)$ is a $\mathcal{I}_{t-1}$-measurable function that defines the filter for the conditional variance and also ensures its positiveness. In the **MSGARCH** package, the initial value of the variance recursions, that is $h_{k,1}$ $(k = 1, \ldots, K)$, are set equal to the unconditional variance in regime k. Depending on the form of $h(\cdot)$, we obtain different scedastic specifications. In the R package **MSGARCH**, we follow this specification in order to reduce model complexity. Finally, when $K = 1$, we recover single-regime GARCH-type models identified by the form of $h(\cdot)$.

[4] The mixture of GARCH model presented in Haas *et al.* (2004b) allows for interactions between the mixture component variances. Here, we report the case referred to as "Diagonal" by Haas *et al.* (2004b).

Below we briefly present the scedastic specifications available in the R package **MSGARCH**. Each of them is identified with a label used in the code for defining a model specification. Similarly, model coefficients are also identified with labels.

ARCH model

The ARCH model of Engle (1982) is given by:

$$h_{k,t} \equiv \alpha_{0,k} + \alpha_{1,k} y_{t-1}^2 \,,$$

for $k = 1, \ldots, K$. In this case, we have $\boldsymbol{\theta}_k = (\alpha_{0,k}, \alpha_{1,k})^\top$. To ensure positivity, we require that $\alpha_{0,k} > 0$ and $\alpha_{1,k} \geq 0$. Covariance-stationarity in each regime is obtained by requiring that $\alpha_{1,k} < 1$. The ARCH specification is identified with the label `"sARCH"`.

GARCH model

The GARCH model of Bollerslev (1986) is given by:

$$h_{k,t} \equiv \alpha_{0,k} + \alpha_{1,k} y_{t-1}^2 + \beta_k h_{k,t-1} \,,$$

for $k = 1, \ldots, K$. In this case, we have $\boldsymbol{\theta}_k = (\alpha_{0,k}, \alpha_{1,k}, \beta_k)^\top$. To ensure positivity, we require that $\alpha_{0,k} > 0$, $\alpha_{1,k} > 0$, and $\beta_k \geq 0$. Covariance-stationarity in each regime is obtained by requiring that $\alpha_{1,k} + \beta_k < 1$. The GARCH specification is identified with the label `"sGARCH"`.

EGARCH model

The Exponential GARCH (EGARCH) of Nelson (1991) is given by:

$$\ln(h_{k,t}) \equiv \alpha_{0,k} + \alpha_{1,k}(|\eta_{k,t-1}| - \mathsf{E}\left[|\eta_{k,t-1}|\right]) + \alpha_{2,k}\eta_{k,t-1} + \beta_k \ln(h_{k,t-1}) \,,$$

for $k = 1, \ldots, K$, where the expectation $\mathsf{E}[|\eta_{k,t-1}|]$ is taken with respect to the distribution conditional on regime k. In this case, we have $\boldsymbol{\theta}_k = (\alpha_{0,k}, \alpha_{1,k}, \alpha_{2,k}, \beta_k)^\top$. This specification takes into consideration the so-called *leverage effect* where past negative observations have a larger influence on the conditional volatility than past positive observations of the same magnitude (Black 1976; Christie 1982). Positivity is automatically ensured by the model specification. Covariance-stationarity in each regime is obtained by requiring that $\beta_k < 1$. The EGARCH specification is identified with the label `"eGARCH"`.

GJR model

The GJR model of Glosten, Jagannathan, and Runkle (1993) is also able to capture the asymmetry in the conditional volatility process. This model is given by:

$$h_{k,t} \equiv \alpha_{0,k} + (\alpha_{1,k} + \alpha_{2,k}\mathbb{I}\{y_{t-1} < 0\}) y_{t-1}^2 + \beta_k h_{k,t-1} \,,$$

for $k = 1, \ldots, K$, where $\mathbb{I}\{\cdot\}$ is the indicator function taking value one if the condition holds, and zero otherwise. In this case, we have $\boldsymbol{\theta}_k = (\alpha_{0,k}, \alpha_{1,k}, \alpha_{2,k}, \beta_k)^\top$. The parameter $\alpha_{2,k}$ controls the degree of asymmetry in the conditional volatility response to the past shock in regime k. To ensure positivity, we require that $\alpha_{0,k} > 0$, $\alpha_{1,k} > 0$, $\alpha_{2,k} \geq 0$, $\beta_k \geq 0$. Covariance-stationarity in each regime is obtained by requiring that $\alpha_{1,k} + \alpha_{2,k}\mathsf{E}[\eta_{k,t}^2 \mathbb{I}\{\eta_{k,t} < 0\}] + \beta_k < 1$. The GJR specification is identified with the label `"gjrGARCH"`.

TGARCH model

Zakoian (1994) introduces the TGARCH specification where the conditional volatility is the dependent variable instead of the conditional variance. This model is given by:

$$h_{k,t}^{1/2} \equiv \alpha_{0,k} + (\alpha_{1,k}\mathbb{I}\{y_{t-1} \geq 0\} - \alpha_{2,k}\mathbb{I}\{y_{t-1} < 0\})y_{t-1} + \beta_k h_{k,t-1}^{1/2}\,,$$

for $k = 1, \ldots, K$. In this case, we have $\boldsymbol{\theta}_k = (\alpha_{0,k}, \alpha_{1,k}, \alpha_{2,k}, \beta_k)^\top$. To ensure positivity, we require that $\alpha_{0,k} > 0$, $\alpha_{1,k} > 0$, $\alpha_{2,k} > 0$ and $\beta_k \geq 0$. Covariance-stationarity in each regime is obtained by requiring that $\alpha_{1,k}^2 + \beta_k^2 - 2\beta_k(\alpha_{1,k} + \alpha_{2,k})\mathsf{E}[\eta_{t,k}\mathbb{I}\{\eta_{t,k} < 0\}] - (\alpha_{1,k}^2 - \alpha_{2,k}^2)\mathsf{E}[\eta_{k,t}^2\mathbb{I}\{\eta_{k,t} < 0\}] < 1$ (see Francq and Zakoian 2011, Section 10.2). The TGARCH specification is identified with the label `"tGARCH"`.

The quantities $\mathsf{E}[\eta_{t,k}\mathbb{I}\{\eta_{t,k} < 0\}]$ and $\mathsf{E}[\eta_{t,k}^2\mathbb{I}\{\eta_{t,k} < 0\}]$ required for the covariance-stationarity conditions in the GJR and TGARCH models, and the quantity $\mathsf{E}\left[|\eta_{k,t-1}|\right]$ required in the conditional variance equation of the EGARCH model, are implemented following Trottier and Ardia (2016).

2.3. Conditional distribution

Model specification is completed by the definition of the conditional distribution of the standardized innovations $\eta_{t,k}$ in each regime of the Markov chain. Here, we present the conditional distributions available in the R package **MSGARCH**. The most common distributions employed to model financial log-returns are implemented; additional distributions might be included in a future release of the R package **MSGARCH** if required by the users. Each distribution is standardized to have a zero mean and a unit variance. As for the conditional variance specification, distributions are identified with labels. Here we drop the time and regime indices for notational purposes, but the shape parameters can be conditional on the regime.

Normal distribution

The probability density function (PDF) of the standard normal distribution is given by:

$$f_N(\eta) \equiv \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}\eta^2}\,, \qquad \eta \in \mathbb{R}\,.$$

This distribution is identified with the label `"norm"`.

Student-t distribution

The PDF of the standardized Student-t distribution is given by:

$$f_S(\eta;\nu) \equiv \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{(\nu-2)\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{\eta^2}{(\nu-2)}\right)^{-\frac{\nu+1}{2}}, \qquad \eta \in \mathbb{R}\,,$$

where $\Gamma(\cdot)$ is the Gamma function. The constraint $\nu > 2$ is imposed to ensure that the second order moment exists. The kurtosis of this distribution is higher for lower ν.[5] This distribution is identified with the label `"std"`.

[5]For $\nu = \infty$, the Student-t distribution is equivalent to the normal distribution.

GED distribution

The PDF of the standardized generalized error distribution (GED) is given by:

$$f_{\text{GED}}(\eta;\nu) \equiv \frac{\nu e^{-\frac{1}{2}|\eta/\lambda|^{\nu}}}{\lambda 2^{(1+1/\nu)}\Gamma(1/\nu)}, \qquad \lambda \equiv \left(\frac{\Gamma(1/\nu)}{4^{1/\nu}\Gamma(3/\nu)}\right)^{1/2}, \qquad \eta \in \mathbb{R},$$

where $\nu > 0$ is the shape parameter.[6] This distribution is identified with the label `"ged"`.

Skewed distributions

Fernández and Steel (1998) provide a simple way to introduce skewness into any unimodal standardized distribution, via the additional parameter $\xi > 0$; if $\xi = 1$ the distribution turns out to be symmetric. Trottier and Ardia (2016) derive the moments of the standardized Fernandez-Steel skewed distributions which are needed in the estimation of the EGARCH, GJR, and TGARCH models. We refer the reader to that publication for details. The skewed version of the normal, Student-t, and GED distributions are identified with the labels `"snorm"`, `"sstd"`, and `"sged"`, respectively.

In Panels A and B of Table 1, we provide a summary of the scedastic functions and conditional distributions available in the package.

2.4. Model estimation

Estimation of MSGARCH and mixture of GARCH models can be done either by maximum likelihood (ML) or by Bayesian Markov chain Monte Carlo (MCMC) techniques. Both approaches require the evaluation of the likelihood function.

Let $\boldsymbol{\Psi} \equiv (\boldsymbol{\theta}_1, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\xi}_K, \mathbf{P})$ be the vector of model parameters. The likelihood function is:

$$\mathcal{L}(\boldsymbol{\Psi} \,|\, \mathcal{I}_T) \equiv \prod_{t=1}^{T} f(y_t \,|\, \boldsymbol{\Psi}, \mathcal{I}_{t-1}), \tag{3}$$

where $f(y_t \,|\, \boldsymbol{\Psi}, \mathcal{I}_{t-1})$ denotes the density of y_t given past observations, $\mathcal{I}_{t-1}$, and model parameters $\boldsymbol{\Psi}$. For MSGARCH, the conditional density of y_t is:

$$f(y_t \,|\, \boldsymbol{\Psi}, \mathcal{I}_{t-1}) \equiv \sum_{i=1}^{K}\sum_{j=1}^{K} p_{i,j}\, z_{i,t-1}\, f_{\mathcal{D}}(y_t \,|\, s_t = j, \boldsymbol{\Psi}, \mathcal{I}_{t-1}), \tag{4}$$

where $z_{i,t-1} \equiv \mathsf{P}[s_{t-1} = i \,|\, \boldsymbol{\Psi}, \mathcal{I}_{t-1}]$ represents the filtered probability of state i at time $t-1$ obtained via Hamilton's filter; see Hamilton (1989) and Hamilton (1994, Chapter 22) for details. Similarly, for the mixture of GARCH model, the conditional density of y_t is:

$$f(y_t \,|\, \boldsymbol{\Psi}, \mathcal{I}_{t-1}) \equiv \sum_{j=1}^{K} \omega_j\, f_{\mathcal{D}}(y_t \,|\, s_t = j, \boldsymbol{\Psi}, \mathcal{I}_{t-1}). \tag{5}$$

In both (4) and (5), the conditional density of y_t in state/component $s_t = k$ given $\boldsymbol{\Psi}$ and $\mathcal{I}_{t-1}$ is denoted by $f_{\mathcal{D}}(y_t \,|\, s_t = k, \boldsymbol{\Psi}, \mathcal{I}_{t-1})$.

[6] Special cases of this distribution are obtained for $\nu = 1$ (Laplace distribution) and $\nu = 2$ (normal distribution). The uniform distribution is obtained in the limit $\nu \to \infty$.

Panel A: Conditional volatility models

Model	Label	Equation
ARCH	`"sARCH"`	$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2$
GARCH	`"sGARCH"`	$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \beta h_{t-1}$
EGARCH	`"eGARCH"`	$\ln(h_t) \equiv \alpha_0 + \alpha_1(\lvert y_{t-1}\rvert - \mathsf{E}[\lvert y_{t-1}\rvert]) + \alpha_2 y_{t-1} + \beta \ln(h_{t-1})$
GJR	`"gjrGARCH"`	$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \alpha_2 y_{t-1}^2 \mathbb{I}\{y_{t-1} < 0\} + \beta h_{t-1}$
TGARCH	`"tGARCH"`	$h_t^{1/2} \equiv \alpha_0 + \alpha_1 y_{t-1}\mathbb{I}\{y_{t-1} \geq 0\} + \alpha_2 y_{t-1}\mathbb{I}\{y_{t-1} < 0\} + \beta h_{t-1}^{1/2}$

Panel B: Conditional distributions

Model	Label	Equation
Normal	`"norm"`	$f_N(\eta) \equiv \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\eta^2}, \eta \in \mathbb{R}$
Student-t	`"std"`	$f_S(\eta;\nu) \equiv \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{(\nu-2)\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{\eta^2}{(\nu-2)}\right)^{-\frac{\nu+1}{2}}, \eta \in \mathbb{R}$
GED	`"ged"`	$f_{\text{GED}}(\eta;\nu) \equiv \frac{\nu e^{-\frac{1}{2}\lvert\eta/\lambda\rvert^\nu}}{\lambda 2^{(1+1/\nu)}\Gamma(1/\nu)}, \lambda \equiv \left(\frac{\Gamma(1/\nu)}{4^{1/\nu}\Gamma(3/\nu)}\right)^{1/2}, \eta \in \mathbb{R}$
Skewed normal	`"snorm"`	See Trottier and Ardia (2016, Equation 1)
Skewed Student-t	`"sstd"`	See Trottier and Ardia (2016, Equation 1)
Skewed GED	`"sged"`	See Trottier and Ardia (2016, Equation 1)

Panel C: Methods

Function	Description
`CreateSpec`	Create a model specification
`DIC`	Compute the deviance information criterion
`ExtractStateFit`	Create a fitted object for each extracted regime
`FitMCMC`	Fit the model by Markov chain Monte Carlo
`FitML`	Fit the model by maximum likelihood
`PIT`	Compute the probability integral transform
`predict`	Compute the conditional volatility forecasts and the density forecasts
`PredPdf`	Compute the predictive density (PDF)
`Risk`	Compute the value-at-risk and expected-shortfall risk measures
`simulate`	Simulate a MSGARCH process
`State`	Compute the in-sample state probabilities
`TransMat`	Compute the transition matrix
`UncVol`	Compute the unconditional volatility
`Volatility`	Compute the in-sample conditional volatilities

Table 1: Specifications and methods available in the R package **MSGARCH**. Panel A reports the conditional volatility models. Panel B reports the PDF of the conditional distributions. Panel C reports the methods.

The ML estimator $\widehat{\boldsymbol{\Psi}}$ is obtained by maximizing the logarithm of (3).[7] In the case of MCMC

[7]Starting values are chosen in the following way: 1) estimate using an expectation-maximization algorithm the static version of the model, that is, with $h_{k,t} = \bar{h}_k$; 2) assign each observation to a regime of the Markov chain using the Viterbi algorithm (see Viterbi 1967) and stack all the series in K vectors, one for each regime; 3) estimate via Quasi-maximum likelihood a volatility model for each vector of decoded observations; 4) estimate via ML the shape parameters of the conditional distribution of the standardized decoded observations. Positivity and covariance-stationarity constraints are guaranteed through specific parameter-mapping functions.

estimation, we follow Ardia (2008), by combining the likelihood with a diffuse (truncated) prior $f(\boldsymbol{\Psi})$ to build the kernel of the posterior distribution $f(\boldsymbol{\Psi} \,|\, \mathcal{I}_T)$. As the posterior is of an unknown form (the normalizing constant is numerically intractable), it must be approximated by simulation techniques. In the R package **MSGARCH**, draws from the posterior are generated with the adaptive random-walk Metropolis sampler of Vihola (2012). For both ML and MCMC estimations, positivity and covariance-stationarity constraints of the conditional variance in each regime are ensured during the estimation.[8]

As detailed in Frühwirth-Schnatter (2006), mixture (and therefore Markov-switching) models are identified up to a relabeling of the coefficients. For MSGARCH and mixture of GARCH models, the same applies when the model specification in each regime is the same, and the prior is symmetric. In the R package **MSGARCH**, by default the identification is carried out by ordering the states according to the unconditional variance of each GARCH-type process, from lower to higher values.

In the case of the Bayesian estimation, the likelihood function is combined with a prior $f(\boldsymbol{\Psi})$ to build the kernel of the posterior distribution $f(\boldsymbol{\Psi} \,|\, \mathcal{I}_T)$. We build our prior from independent diffuse priors as follows:

$$
\begin{aligned}
f(\boldsymbol{\Psi}) &= f(\boldsymbol{\theta}_1, \boldsymbol{\xi}_1) \cdots f(\boldsymbol{\theta}_K, \boldsymbol{\xi}_K) f(\mathbf{P}) \\
f(\boldsymbol{\theta}_k, \boldsymbol{\xi}_k) &\propto f(\boldsymbol{\theta}_k) f(\boldsymbol{\xi}_k) \, \mathbb{I}\{(\boldsymbol{\theta}_k, \boldsymbol{\xi}_k) \in \mathcal{CSC}_k\} \quad (k = 1, \ldots, K) \\
f(\boldsymbol{\theta}_k) &\propto f_{\mathcal{N}}(\boldsymbol{\theta}_k; \boldsymbol{\mu}_{\boldsymbol{\theta}_k}, \operatorname{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}_k})) \, \mathbb{I}\{\boldsymbol{\theta}_k \in \mathcal{PC}_k\} \quad (k = 1, \ldots, K) \\
f(\boldsymbol{\xi}_k) &\propto f_{\mathcal{N}}(\boldsymbol{\xi}_k; \boldsymbol{\mu}_{\boldsymbol{\xi}_k}, \operatorname{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\xi}_k})) \, \mathbb{I}\{\xi_{k,1} > 0, \xi_{k,2} > 2\} \quad (k = 1, \ldots, K) \\
f(\mathbf{P}) &\propto \prod_{i=1}^{K} \left(\prod_{j=1}^{K} p_{i,j} \right) \mathbb{I}\{0 < p_{i,i} < 1\},
\end{aligned}
\tag{6}
$$

where $\mathcal{CSC}_k$ denotes the covariance-stationarity condition and $\mathcal{PC}_k$ the positivity condition in regime k; see Trottier and Ardia (2016). $\xi_{k,1}$ is the asymmetry parameter and $\xi_{k,2}$ the tail parameter of the skewed Student-t distribution in regime k. $f_{\mathcal{N}}(\bullet; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate normal density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Finally, $\boldsymbol{\mu}_\bullet$ and $\boldsymbol{\sigma}^2_\bullet$ are vectors of prior means and variances (of appropriate sizes), whose entries are set by default to zero and to 1,000, respectively.

3. The R package MSGARCH

3.1. Model specification

Model specification in the R package **MSGARCH** is performed via the `CreateSpec()` function. `CreateSpec()` accepts several arguments and returns an S3 object of class '`MSGARCH_SPEC`' for which methods such as `print()` and `summary()` (among others) are available; we refer the reader to `help("CreateSpec")` for the complete documentation. The arguments of `CreateSpec()` are:

[8]Note that these constraints are stronger than those derived by Haas *et al.* (2004a). However, they allow us to introduce a wider range of conditional variance specifications and distributional assumptions in the **MSGARCH** package.

- `variance.spec`: A list with the element `model`, a `character` vector (of size K, that is, the number of regimes/components) containing the conditional variance specifications. Valid models are `"sARCH"`, `"sGARCH"`, `"eGARCH"`, `"gjrGARCH"` and `"tGARCH"` (see Section 2.2 for details). By default, `variance.spec = list(model = c("sGARCH", "sGARCH"))`.

- `distribution.spec`: A list with element `distribution`, a `character` vector (of size K) containing the conditional distribution specifications. Valid distributions are `"norm"`, `"snorm"`, `"std"`, `"sstd"`, `"ged"`, and `"sged"` (see Section 2.3 for details). The length of `distribution` has to be the same as the length of `model`, otherwise an error is reported. By default, `distribution.spec = list(distribution = c("norm", "norm"))`.

- `switch.spec`: A list with elements `do.mix` and `K`. The first element, `do.mix`, is a logical indicating if the model specification is Markov-switching or mixture of GARCH. If `do.mix = TRUE`, a mixture of GARCH (Haas *et al.* 2004b) is specified, while if `do.mix = FALSE`, a MSGARCH (Haas *et al.* 2004a) is specified. By default `do.mix = FALSE`. The second element, `K`, is an `integer` which controls for the number of regime/components. If the length of `model` in `variance.spec` and `distribution` in `distribution.spec` is one, then the chosen specification is the same across the K regime/components. By default, `K = NULL`, that is, the number of components is determined from the length of the vector `model` in `variance.spec`.

- `constraint.spec`: A list with elements `fixed` and `regime.const`. The first element, `fixed`, is a list with numeric entries and named elements. This argument controls for fixed parameters set by the user. The names of the `fixed` entries have to coincide with the labels associated to the model parameters. For instance, if `fixed = list(beta_1 = 0))`, `beta_1` will be fixed to `0` during the optimization. The second element, `regime.const`, is a `character` vector. It controls for the parameters which are set equal across regimes. The names of the entries in the list have to coincide with the names of the model parameters minus the regime indicator. For instance, if `constraint.spec = list(regime.const = "beta")`, all the parameters named `beta` will be the same across regimes. Note that both types of constraint cannot be defined contemporaneously, that is, either `fixed` or `regime.const` can be specified. By default, `fixed` and `regime.const` are set to `NULL`.

- `prior`: A list with elements `mean` and `sd`. The elements `mean` and `sd` are both lists with numeric named elements which allow the user to adjust the prior mean and standard deviation of the independent (symmetric) truncated normal priors in (6). The names of the entries in the lists have to coincide with the labels associated to model parameters. For instance, if `prior = list(mean = list(beta_1 = 0.7), sd = list(beta_1 = 0.1))`, the prior mean of `beta_1` will be set to `0.7` while the prior standard deviation will be set to `0.1`.

As an illustration, let us consider the MSGARCH model of Haas *et al.* (2004a) with $K = 2$ regimes. This model assumes that conditionally on each regime of the Markov chain, returns are normally distributed with GARCH$(1, 1)$ variances. It can be easily specified with the following lines of code:

```
R> library("MSGARCH")
R> spec <- CreateSpec()
```

The relevant information is summarized with the `summary` method:

```
R> summary(spec)

Specification type: Markov-switching
Specification name: sGARCH_norm sGARCH_norm
Number of parameters in each variance model: 3 3
Number of parameters in each distribution: 0 0
----------------------------------------
Fixed parameters:
None
----------------------------------------
Across regime constrained parameters:
None
----------------------------------------
```

The output printed in the console provides information regarding: (i) the model, such as whether a Markov-switching or mixture of GARCH has been selected, (ii) the GARCH-type specification within each regime, (iii) the number of variance parameters, and (iv) the number of shape parameters in each regime. The presence of fixed parameters or equal parameters across regimes set by `fixed.pars` and `regime.const.pars` is also displayed.

We now report examples of various MSGARCH models that can be specified in the R package **MSGARCH**. We refer the reader to the documentation for additional examples; see `help("CreateSpec")`.

Example 1: A single-regime model

The R package **MSGARCH** also supports single-regime models, as they are the building blocks of Markov-switching models. A simple specification is the GARCH model with normal conditional distribution:

```
R> spec <- CreateSpec(variance.spec = list(model = "sGARCH"),
+    distribution.spec = list(distribution = "norm"))
```

Example 2: A model with heterogeneous regimes

The R package **MSGARCH** has a modular approach to define the conditional variance and density in each regime, the state dynamics and the number of regimes. Thanks to this flexibility, the user can choose many combinations of specifications. Here we report an example of a three-state MSGARCH model, where each regime is characterized by a different conditional volatility and a different conditional distribution:

```
R> spec <- CreateSpec(
+    variance.spec = list(model = c("sGARCH", "tGARCH", "eGARCH")),
+    distribution.spec = list(distribution = c("snorm", "std", "sged")))
```

Example 3: A model with non-switching shape parameters

The user can also choose to constrain the parameters to be the same across regimes. In the R package **MSGARCH** package, the shape parameters ν and ξ reported in Section 2.3 are identified with the labels `"nu"` and `"xi"`, respectively. Here is an example of a two-state MSGARCH model where the conditional distributions' shape parameters of both regimes are constrained to be the same:

```
R> spec <- CreateSpec(variance.spec = list(model = c("sGARCH", "sGARCH")),
+    distribution.spec = list(distribution = c("sstd", "sstd")),
+    constraint.spec = list(regime.const = c("nu", "xi")))
```

In this model, the only element that switches according to the Markov chain is the conditional variance.

3.2. Model estimation

In the R package **MSGARCH**, model estimation can be either achieved by maximum likelihood (ML) using the function `FitML()`, or via Markov chain Monte Carlo (MCMC) simulation using the function `FitMCMC()`.[9] These functions accept three common arguments which are: (i) `spec`, (ii) `data`, and (iii) `ctr`. The first argument, `spec`, is an '`MSGARCH_SPEC`' object created with the `CreateSpec()` function detailed in Section 3.1. The second argument, `data`, is a numeric vector of T observations. The last argument, `ctr`, is a list of control parameters for model estimation.

Control parameters are different between the `FitML()` and `FitMCMC()` functions. Regarding the function `FitML()`, the following controls may be defined:

- `par0`: A numeric vector of starting parameters that overwrites the default starting parameter scheme (see footnote 7 on page 8). The starting parameters should follow the order of the default parameter of the '`MSGARCH_SPEC`' object. By default `par0 = NULL`.
- `do.se`: A logical indicating if standard errors are computed. In the case where the standard errors are not needed, setting `do.se = FALSE` will speed up estimation. Default is `do.se = TRUE`.
- `OptimFUN`: A custom optimization function set by the user. By default, `OptimFUN` is set such that optimization is done via the well-known Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm using the `optim()` function with `method = "BFGS"`. For a custom optimizer, we refer to the examples in `help("FitML")`.

The function `FitML()` outputs an object of class '`MSGARCH_ML_FIT`' which can be used with several **MSGARCH** functionalities (see Section 3.5). We refer to `help("FitML")` for more details.

Regarding the function `FitMCMC()`, the following control parameters may be defined:

[9] We follow Ardia (2008) and use non-informative truncated normal priors.

- `par0`: A numeric vector of starting parameters, as for `FitML()`.
- `nburn`: An `integer` indicating the number of discarded draws (i.e., the burn-in phase). Default is `nburn = 500L`.
- `nmcmc`: An `integer` indicating the number of MCMC draws. Default is `nmcmc = 10000L`.
- `nthin`: An `integer` indicating the thinning factor (every `nthin` draws are kept in the posterior sample). Default is `nthin = 10L`.
- `SamplerFUN`: A custom MCMC sampler. By default, `SamplerFUN` is set up such that the estimation relies on the adaptive Metropolis-Hastings sampler described in Vihola (2012). A custom MCMC sampler can be set up by the user using the `SamplerFUN` element of `ctr`. We refer to the example section of `help("FitMCMC")` for an illustration.
- `do.sort`: A logical indicating if the MCMC draws are post-processed and sorted following Geweke (2007). By default, `do.sort = TRUE`, such that the MCMC draws are ordered to ensure that the unconditional variance is an increasing function of the state value. If the user sets `do.sort = FALSE`, no sorting is imposed, and label switching can occur (for a given model specification). In this case, the user can extract the MCMC chain and investigate a suitable identification constraint with the R package **label.switching** (Papastamoulis 2016).[10]

The main purpose of `nthin` is to diminish the autocorrelation in the MCMC chain. The argument `nburn` also serves as pre-optimization step. The total length of the chain is: `nmcmc`/`nthin`. Finally, the chain is converted to a **coda** object, meaning that all functions for MCMC analysis available in the R package **coda** (Plummer, Best, Cowles, and Vines 2006) are available.

Illustration of `FitML()`

As an example, we estimate the MSGARCH model of Haas *et al.* (2004a) on daily observations of the Deutschmark vs. British Pound (DEM/GBP) foreign exchange log-returns available in the **MSGARCH** package. The sample period is from January 3, 1985, to December 31, 1991, for a total of 1,974 observations. This data set has been promoted as a standard benchmark for GARCH time series software validation (Brooks, Burke, and Persand 2001). It can be loaded in the workspace by running:

```
R> data("dem2gbp", package = "MSGARCH")
```

We then fit the model to the `dem2gbp` data set using:

```
R> ms2.garch.n <- CreateSpec(variance.spec = list(model = "sGARCH"),
+    distribution.spec = list(distribution = "norm"),
+    switch.spec = list(K = 2))
R> fit.ml <- FitML(spec = ms2.garch.n, data = dem2gbp)
R> summary(fit.ml)
```

[10]Since our MCMC sampler does not generate draws for the state variable, only the methods `aic`, `pra`, `stephens`, and `sjw` in the R package **label.switching** can be used.

```
Specification type: Markov-switching
Specification name: sGARCH_norm sGARCH_norm
Number of parameters in each variance model: 3 3
Number of parameters in each distribution: 0 0
----------------------------------------
Fixed parameters:
None
----------------------------------------
Across regime constrained parameters:
None
----------------------------------------
Fitted parameters:
         Estimate Std. Error t value  Pr(>|t|)
alpha0_1   0.0007     0.0003   2.307 1.053e-02
alpha1_1   0.0515     0.0184   2.796 2.590e-03
beta_1     0.9178     0.0061 149.819    <1e-16
alpha0_2   0.2813     0.1495   1.882 2.994e-02
alpha1_2   0.4805     0.4764   1.008 1.566e-01
beta_2     0.3996     0.2202   1.815 3.479e-02
P_1_1      0.9109     0.0680  13.394    <1e-16
P_2_1      0.5947     0.0369  16.127    <1e-16
----------------------------------------
Transition matrix:
      t+1|k=1 t+1|k=2
t|k=1  0.9109  0.0891
t|k=2  0.5947  0.4053
----------------------------------------
Stable probabilities:
State 1 State 2
 0.8697  0.1303
----------------------------------------
LL: -971.911
AIC: 1959.822
BIC: 2004.5245
----------------------------------------
```

The output of `summary()` reports various information regarding model estimation. Estimated parameters are displayed along with significance levels according to their asymptotic Gaussian distribution. The summary also returns the unconditional distribution (`Stable probabilities`) of the Markov chain, $\boldsymbol{\pi}$, such that $\mathbf{P}\boldsymbol{\pi} = \boldsymbol{\pi}$. Finally, the log-likelihood evaluated at its optimum together with Akaike and Bayesian information criteria are also reported. Note that the ML estimation results are ordered with respect to the unconditional variance in each regime, from lower to higher values, when regimes have the same model specification.

Illustration of `FitMCMC()`

The function `FitMCMC()` outputs an object of class 'MSGARCH_MCMC_FIT' which, like the class 'MSGARCH_ML_FIT', can be used with several **MSGARCH** functionalities. We refer the reader to the documentation for details; see `help("FitMCMC")`.

Bayesian estimation for the MSGARCH model previously specified in the object `ms2.garch.n` is easily performed with:

```
R> set.seed(1234)
R> fit.mcmc <- FitMCMC(spec = ms2.garch.n, data = dem2gbp)
```

The adaptive MCMC estimation procedure generates draws from the posterior distribution.[11] These draws, referred to as the posterior sample, are used to characterize the distribution of the parameters. As for `FitML()`, the `summary()` method is defined for an object of class 'MSGARCH_MCMC_FIT' delivered by `FitMCMC()`:

```
R> summary(fit.mcmc)

Specification type: Markov-switching
Specification name: sGARCH_norm sGARCH_norm
Number of parameters in each variance model: 3 3
Number of parameters in each distribution: 0 0
-----------------------------------------
Fixed parameters:
None
-----------------------------------------
Across regime constrained parameters:
None
-----------------------------------------
Posterior sample (size: 1000)
           Mean     SD     SE   TSSE    RNE
alpha0_1 0.0011 0.0003 0.0000 0.0000 0.5000
alpha1_1 0.0315 0.0074 0.0002 0.0003 0.4944
beta_1   0.9401 0.0118 0.0004 0.0005 0.4751
alpha0_2 0.6341 0.1200 0.0038 0.0061 0.3807
alpha1_2 0.0756 0.0475 0.0015 0.0034 0.1900
beta_2   0.0431 0.0680 0.0022 0.0043 0.2535
P_1_1    0.9565 0.0095 0.0003 0.0004 0.5135
P_2_1    0.1785 0.0286 0.0009 0.0018 0.2500
-----------------------------------------
Posterior mean transition matrix:
      t+1|k=1 t+1|k=2
t|k=1  0.9565  0.0435
t|k=2  0.1785  0.8215
```

[11] The adaptive MCMC sampler requires Cholesky decomposition and eigenvalue calculation. The obtained results, therefore, depend on the linear algebra library used. We refer the reader to the computational details at the end of the paper for more discussion about this.

```
-----------------------------------------
Posterior mean stable probabilities:
State 1 State 2
 0.8042  0.1958
-----------------------------------------
Acceptance rate MCMC sampler: 27.6%
nmcmc: 10000
nburn: 5000
nthin: 10
-----------------------------------------
DIC: 1986.655
-----------------------------------------
```

Among the various information reported, part of the output is generated by calling the `summary()` method of the R package **coda** (Plummer *et al.* 2006). This part coincides with: (i) the posterior mean (`Mean`), (ii) standard deviation (`SD`), (iii) naive standard error of the mean (i.e., ignoring the autocorrelation in the MCMC chain) (`SE`), (iv) time series standard error based on an estimate of the spectral density at zero (`TSSE`), and (v) the relative numerical efficiency (`RNE`), defined as $(\texttt{SE}/\texttt{TSSE})^2$. Other useful statistics are the acceptance rate of the MCMC sampler and the deviance information criterion (DIC) of Spiegelhalter, Best, Carlin, and Van der Linde (2002).

3.3. Forecasting

The R package **MSGARCH** implements functionalities to perform forecasting of the underlying time series for which the model has been estimated. Suppose an MSGARCH model is estimated using the first T observations $(y_1, \ldots, y_T)$. Then, the user may be interested in: (i) forecasting the volatility at time $T+h$ $(h > 0)$, that is, the standard deviation of the random variable $y_{T+h}|\mathcal{I}_T$; or (ii) forecasting the shape of the distribution of $y_{T+h}|\mathcal{I}_T$. Both types of forecasts can be easily obtained using the function `predict()` in the R package **MSGARCH** starting from an object of class '`MSGARCH_MLE_FIT`' or '`MSGARCH_BAY_FIT`' obtained via the functions `FitML()` and `FitMCMC()`, respectively.

Volatility and density forecasts are performed via the `predict()` method defined for an object of class '`MSGARCH_MLE_FIT`' or '`MSGARCH_BAY_FIT`'. The horizon of the prediction, h, is determined with the `integer` argument `nahead` (default: `nahead = 1L`). For $h = 1$, the predictive volatility and density are available in closed form; in other circumstances (if $h > 1$) forecasts are obtained by simulation, drawing iteratively a new observation from the (one-step ahead) predictive distribution and updating the K conditional variance processes accordingly. The number of draws controlling for the accuracy of the simulations are defined in the `ctr` argument. Specifically, the entry `nsim` in `ctr` defines the number of draws such that, `ctr = list(nsim = 1e4L)` indicates that 10,000 simulations are performed.

If an object of class '`MSGARCH_BAY_FIT`' is provided, the number of simulations used to draw from the predictive distribution is equal to the number of draws in the posterior distributions.[12] However, if `nsim` is specified, then for each posterior draw, `nsim` observations are sampled.

[12] This is specified in the `ctr` argument of the `FitMCMC()` function via the `nmcmc` argument.

The `predict()` method returns an object of class 'MSGARCH_FORECAST' with two elements: `vol` and `draw`. The first element, `vol`, is a numeric vector of length h, containing the standard deviations of the distributions $y_{T+j}|\mathcal{I}_T$ for $j = 1, \ldots, h$. The second element, `draw`, is a `matrix` of dimension $h \times$ `nsim` of the simulated MSGARCH process. If we set `do.return.draw = FALSE` in the function call, then `draw` is `NULL`.

For instance, the following code returns the five-step ahead conditional volatilities and draws for the `dem2gbp` log-returns data using the MSGARCH model of Haas *et al.* (2004a) estimated in Section 3.2:

```
R> pred <- predict(fit.ml, nahead = 5, do.return.draw = TRUE)
R> pred$vol

   h=1    h=2    h=3    h=4    h=5
0.4006 0.3967 0.3836 0.3938 0.4004

R> pred$draw[, 1:4]

      Sim #1   Sim #2  Sim #3    Sim #4
h=1  0.27169  0.39881  0.3893 -0.185686
h=2 -0.04223 -0.08693 -0.1556 -0.083054
h=3 -0.03076 -0.55994  0.2884 -0.060956
h=4  0.00159  0.27208 -0.1725  0.086989
h=5 -0.18180 -0.06522 -1.6092  0.004029
```

The `predict()` method also accepts the additional numeric argument `newdata` which allows us to use a previous fit and to augment the data with new observations before forecasting. Thus, if the vector of new observations is of size T^*, the prediction is made for the random variable $y_{T+T^*+h}|\mathcal{I}_{T+T^*}$, using the parameter estimates obtained with observations up to time T.

3.4. Quantitative risk management

One of the main applications of MSGARCH models is in quantitative finance where investors want to allocate their wealth among a series of risky investment opportunities. In this area, observations usually coincide with log-returns of financial assets, and quantities of interest include the quantile of their future distribution at a specific risk level $\alpha \in (0, 1)$ as well as their expected value below this level. These two quantities are referred to as value-at-risk (VaR) and expected-shortfall (ES), respectively (see, e.g., McNeil *et al.* 2015). The VaR measures the threshold value such that the probability of observing a loss larger or equal to it in a given time horizon is equal to α. The ES measures the expected loss below the VaR level.

Formally, the VaR forecast in $T + 1$ at risk level α (given the information set up to at time T) is defined as:

$$\text{VaR}^{\alpha}_{T+1} \equiv \inf \left\{ y_{T+1} \in \mathbb{R} \,|\, F(y_{T+1} \,|\, \mathcal{I}_T) = \alpha \right\} \,, \tag{7}$$

where $F(y|\mathcal{I}_T)$ is the one-step ahead cumulative density function (CDF) evaluated in y. The ES is obtained as:

$$\text{ES}^{\alpha}_{T+1} \equiv \mathsf{E}[y_{T+1} \,|\, y_{T+1} \leq \text{VaR}^{\alpha}_{T+1}, \mathcal{I}_T] \,, \tag{8}$$

It is straightforward to estimate these two metrics with the R package **MSGARCH**, starting from a fitted object of class 'MSGARCH_ML_FIT' or 'MSGARCH_MCMC_FIT'. Evaluation of VaR and ES can be performed both in-sample and out-of-sample using the `Risk()` function. In-sample evaluation means that we want to investigate the evolution of the two risk measures over the estimation period, whereas out-of-sample refers to prediction. The α level is controlled via the `alpha` argument in `Risk()`. The argument `alpha` is a numeric vector such that different α levels can be evaluated contemporaneously. By default `alpha = c(0.01, 0.05)`. Other arguments are:

- `do.es`: A logical indicating if expected-shortfall is also calculated. By default, `do.es = TRUE`, such that both VaR and ES are reported. (Note that `do.es = FALSE` speeds up the computations.)
- `do.its`: A logical indicating if in-sample risk measures are computed. Default is `do.its = FALSE` such that out-of-sample risk measures are computed.
- `newdata`: A numeric vector of new observations (see Section 3.3). By default, `newdata = NULL`.
- `nahead`: An `integer` indicating the forecast horizon. By default, `nahead = 1L`.
- `do.cumulative`: A logical indicating if the risk measure should be computed on the distribution of simulated cumulative values. This is useful in the case of log-returns, to compute the risk measures for the distribution of aggregated returns. By default, `do.cumulative = FALSE`.
- `ctr`: A list of control parameters; see `help("Risk")`.

The function `Risk()` creates an object of class 'MSGARCH_RISK' with elements `VaR` and `ES`. Both `VaR` and `ES` are a `matrix` of dimension `nahead` $\times$ `length(alpha)` containing the risk measures, VaR and ES, respectively. If `do.its = TRUE`, these matrices contain the in-sample levels such that their dimension is $T \times$ `length(alpha)`.[13]

For instance, using the `fit.ml` object created in Section 3.3, the five-step ahead VaR and ES measures at the 1% and 5% risk levels are computed as follows:

```
R> risk <- Risk(fit.ml, alpha = c(0.01, 0.05), nahead = 5)
R> risk$VaR

      0.01    0.05
h=1 -1.211 -0.5851
h=2 -1.153 -0.5403
h=3 -1.101 -0.5469
h=4 -1.166 -0.5641
h=5 -1.085 -0.5614

R> risk$ES
```

[13] If `newdata` is provided, the dimension is $(T + T^*) \times$ `length(alpha)`, where, as before, T^* = `length(newdata)`.

```
      0.01    0.05
h=1 -1.549 -0.9403
h=2 -1.556 -0.9024
h=3 -1.588 -0.9075
h=4 -1.587 -0.9265
h=5 -1.551 -0.9068
```

Note that, if model estimation is performed via MCMC, and hence an object of class 'MSGARCH_MCMC_FIT' is provided to `Risk()`, the VaR and ES measures integrate the parameter uncertainty; see Hoogerheide and Van Dijk (2010).

3.5. Other functionalities

Several other functionalities are available in the R package **MSGARCH**, which allow the user to extract the in-sample conditional volatility and latent states (functions `Volatility()` and `State()`, respectively), to simulate (function `simulate()`), to compute the predictive density (function `PredPdf()`) and the probability integral transform (function `PIT()`). We refer the reader to the documentation manual for details; see `help("MSGARCH")`.

In all cases, objects of classes 'MSGARCH_ML_FIT' and 'MSGARCH_MCMC_FIT' can be used as an input. In the case of the MCMC estimation, all functions return the aggregated value (using the mean) over MCMC draws, thus integrating the parameter uncertainty.

For instance, we can easily simulate from the previous estimated model with `FitML()`. Below, we generate two paths of length four with a burn-in phase of length 500.

```
R> simulate(fit.ml, nsim = 2, nahead = 4, nburn = 500)

$draw
     Sim #1 Sim #2
t=1 -0.1322 0.1943
t=2 -0.5581 0.2910
t=3 -0.5703 0.6642
t=4 -0.3568 0.2495

$state
    Sim #1 Sim #2
t=1      1      2
t=2      1      2
t=3      2      2
t=4      2      1

$CondVol
, , k=1

    Sim #1 Sim #2
t=1 0.2962 0.4212
t=2 0.2866 0.4067
t=3 0.3035 0.3961
```

```
t=4 0.3193 0.4091

, , k=2

    Sim #1 Sim #2
t=1 0.7095 0.7025
t=2 0.7006 0.7047
t=3 0.7919 0.7214
t=4 0.8295 0.8374
```

The function outputs the simulated MSGARCH process together with the simulated regimes and the conditional volatilities in each state.

It is worth emphasizing the difference between `predict()` and `simulate()`. The function `predict()` is used for forecasting the conditional volatility and the predictive distribution (using the argument `do.return.draw = TRUE`) while the function `simulate()` aims at generating simulation paths for a given MSGARCH model.

To perform in-sample model selection, information criteria such as the Akaike information criterion (Akaike 1974), the Bayesian information criterion (Schwarz 1978), and the deviance information criterion (Spiegelhalter *et al.* 2002) are available. These are all measures of the parsimony of statistical models in describing a given set of data, where lower values indicate a better performance in terms of goodness-of-fit. They are estimated with the functions `AIC()` and `BIC()` from the **stats** package (R Core Team 2018), and `DIC()` from the **MSGARCH** package, respectively.

For instance, the BIC can be computed from the model fit:

```
R> BIC(fit.ml)

[1] 2005
```

Finally, the function `ExtractStateFit()` allows the user to extract single-regime model results from the fitted object '`MSGARCH_ML_FIT`' or '`MSGARCH_MCMC_FIT`'. Hence, if the user estimates a model with K regimes, he can retrieve the information of the K single-regime layers as a list of length K. For instance, still considering the object `fit.ml`, if we want to predict the five-step ahead VaR conditionally of being in the two regimes, we simply write:

```
R> sr.fit <- ExtractStateFit(fit.ml)
R> risk1 <- Risk(sr.fit[[1]], alpha = 0.05, nahead = 5)
R> risk2 <- Risk(sr.fit[[2]], alpha = 0.05, nahead = 5)
R> VaR <- cbind(risk1$VaR, risk2$VaR)
R> colnames(VaR) <- c("State 1", "State 2")
R> VaR

    State 1 State 2
h=1 -0.4411  -1.293
h=2 -0.4314  -1.515
h=3 -0.4244  -1.633
h=4 -0.4322  -1.724
h=5 -0.4206  -1.742
```

Hence, we are able to evaluate the risk exposure of an investment conditionally on different regimes of the market. All the other package functionalities, like `Volatility()` and `predict()`, can be applied to the single-regime layers extracted via `ExtractStateFit()`. This way, scenario analyses can be easily performed.

In Panel C of Table 1, we provide a summary of the various methods available in the package.

4. Empirical illustration

In this section, we illustrate how the R package **MSGARCH** can be used for model comparison, state/regime smoothing, and volatility filtering. Estimation via MCMC is also discussed. Our illustration focuses on an in-sample analysis of the daily log-returns of the major equity index for the Swiss market, namely the Swiss market index (SMI). MSGARCH models have been shown to be superior than single-regime counterparts on out-of-sample backtesting results; see Ardia, Bluteau, Boudt, and Catania (2018) for a large-scale empirical study of their performance. An example of a backtest implementation is provided at the end of this section.

The dataset can be loaded in the workspace by running `data("SMI", package = "MSGARCH")` in the console. The plot of the time series is presented in Figure 1. Well-known stylized facts observed in financial time series, such as volatility clustering and presence of outliers, are evident from Figure 1 (see, e.g., McNeil *et al.* 2015). Furthermore, we also note that large (absolute) returns are more frequent at the start (1990–1993) and at the end (1997–2000) of

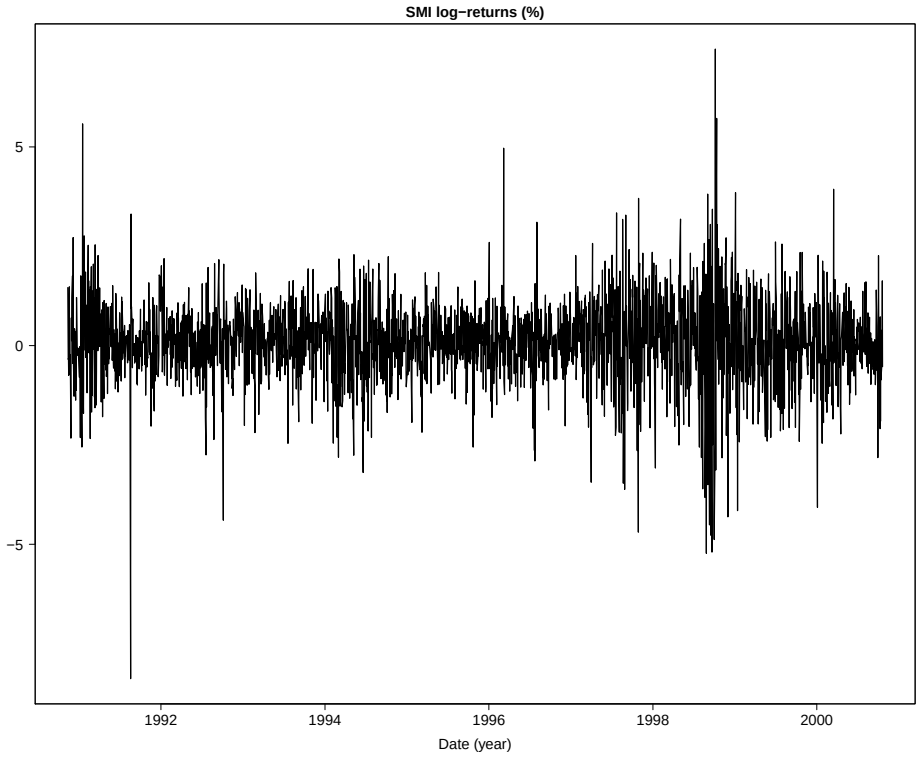

Figure 1: Percentage daily log-returns of the Swiss market index for a period ranging from November 12, 1990, to October 20, 2000, for a total of 2,500 observations.

the sample, than in the middle (1993–1997). This suggests that the conditional variance is time-varying according to a regime-switching specification. For the sake of illustration, we consider the asymmetric two-state MSGARCH model implemented by Ardia (2008, Chapter 7) and Mullen, Ardia, Gil, Windover, and Cline (2011, Section 5). This is an extension of the MSGARCH model introduced in Haas *et al.* (2004a), where a GJR variance specification with a Student-t distribution is assumed in each regime. The model may be written as:

$$\begin{aligned} y_t \,|\, (s_t = k, \mathcal{I}_{t-1}) &\sim S(0, h_{k,t}, \nu)\,, \\ h_{k,t} &\equiv \alpha_{0,k} + (\alpha_{1,k} + \alpha_{2,k}\mathbb{I}\{y_{t-1} < 0\})y_{t-1}^2 + \beta_k h_{k,t-1}\,, \end{aligned} \tag{9}$$

where $k \in \{1, 2\}$. Note that the degree of freedom parameter of the Student-t distribution is fixed across the regimes. Model specification and ML estimation of this model is performed with the following code:

```
R> ms2.gjr.s <- CreateSpec(variance.spec = list(model = "gjrGARCH"),
+    distribution.spec = list(distribution = "std"),
+    switch.spec = list(K = 2),
+    constraint.spec = list(regime.const = "nu"))
R> fit.ml <- FitML(ms2.gjr.s, data = SMI)
```

The summary of the estimation is obtained as follows:

```
R> summary(fit.ml)
```

```
Specification type: Markov-switching
Specification name: gjrGARCH_std gjrGARCH_std
Number of parameters in each variance model: 4 4
Number of parameters in each distribution: 1 1
----------------------------------------
Fixed parameters:
None
----------------------------------------
Across regime constrained parameters:
nu
----------------------------------------
Fitted parameters:
         Estimate Std. Error  t value  Pr(>|t|)
alpha0_1   0.2071     0.0488   4.2432 1.102e-05
alpha1_1   0.0005     0.0088   0.0569 4.773e-01
alpha2_1   0.2137     0.0619   3.4505 2.798e-04
beta_1     0.5264     0.0995   5.2921 6.045e-08
nu_1       9.2468     1.3292   6.9569 1.739e-12
alpha0_2   0.0922     0.0349   2.6397 4.149e-03
alpha1_2   0.0052     0.0169   0.3050 3.802e-01
alpha2_2   0.1516     0.0381   3.9771 3.488e-05
beta_2     0.8716     0.0354  24.6380    <1e-16
P_1_1      0.9977     0.0014 701.8429    <1e-16
```

```
P_2_1      0.0027     0.0017   1.6006 5.473e-02
----------------------------------------
Transition matrix:
      t+1|k=1 t+1|k=2
t|k=1  0.9977  0.0023
t|k=2  0.0027  0.9973
----------------------------------------
Stable probabilities:
State 1 State 2
 0.5407  0.4593
----------------------------------------
LL: -3350.8467
AIC: 6723.6934
BIC: 6787.7579
----------------------------------------
```

Parameter estimates indicate that the evolution of the volatility process is heterogeneous across the two regimes. Indeed, we first note that the two regimes report different unconditional volatility levels:

```
R> set.seed(1234)
R> sqrt(250) * sapply(ExtractStateFit(fit.ml), UncVol)

[1] 11.87 22.04
```

as well as a different reactions to past negative returns: $\alpha_{2,1} \approx 0.21$ vs. $\alpha_{2,2} \approx 0.15$. Also the volatility persistence in the two regimes is different. The first regime reports $\alpha_{1,1}+\frac{1}{2}\alpha_{2,1}+\beta_1 \approx 0.63$ while the second regime reports $\alpha_{1,2}+\frac{1}{2}\alpha_{2,2}+\beta_2 \approx 0.95$.[14] In summary, the first regime is characterized by: (i) low unconditional volatility, (ii) strong volatility reaction to past negative returns, and (iii) low persistence of the volatility process. Differently, the second regime is characterized by: (i) high unconditional volatility, (ii) weak volatility reaction to past negative returns, and (iii) high persistence of the volatility process. Clearly, regime one would be perceived by market participants as "tranquil market conditions" with low volatility levels, low persistence and high reaction to past negative returns, while regime two as "turbulent market conditions" with high volatility levels and strong persistence.

Filtered, predicted, and smoothed probabilities (i.e., $\mathsf{P}[S_t = k|\widehat{\boldsymbol{\Psi}}, \mathcal{I}_t]$, $\mathsf{P}[S_t = k|\widehat{\boldsymbol{\Psi}}, \mathcal{I}_{t-1}]$ and $\mathsf{P}[S_t = k|\widehat{\boldsymbol{\Psi}}, \mathcal{I}_T]$, respectively) can be computed starting from estimated objects using the `State()` function. `State()` reports a list of four elements: (i) `FiltProb`, (ii) `PredProb`, (iii) `SmoothProb`, and (iv) `Viterbi`. The first three elements are `array` objects of dimension $T \times 1 \times K$ containing the filtered, predicted, and smoothed probabilities at each time t, respectively. The last element, `Viterbi`, is a `matrix` of dimension $T \times 1$ of indices representing decoded states according to the Viterbi algorithm detailed in Viterbi (1967).[15] For instance, smoothed probabilities of being in the second regime ($k = 2$) can be computed with the following code:

[14] Recall that $\mathsf{E}[\eta^2 \mathbb{I}\{\eta < 0\}] = \frac{1}{2}$ when η is symmetrically distributed.

[15] The reason why `State()` outputs arrays of dimension $T \times 1 \times K$ is because when it is evaluated for an object of class '`MSGARCH_MCMC_FIT`', the second dimension corresponds to the number of posterior draws. Analogously for the `Viterbi matrix`.

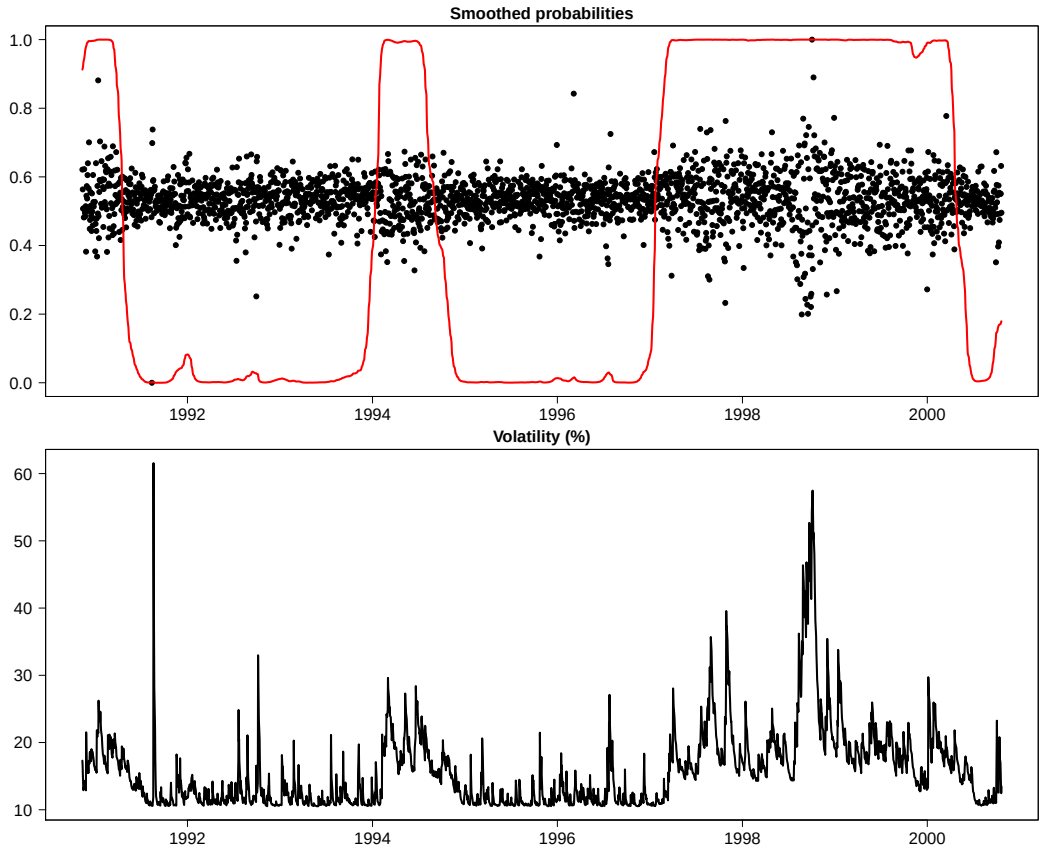

Figure 2: Top: Estimated smoothed probabilities of the second regime, $\mathsf{P}[s_t = 2|\widehat{\boldsymbol{\Psi}}, \mathcal{I}_T]$, for $t = 1, \ldots, T$. The thin black line depicts the SMI log-returns. Bottom: Filtered conditional volatilities.

```
R> smooth.prob <- State(fit.ml)$SmoothProb[, 1, 2, drop = TRUE]
```

Figure 2 displays the smoothed probabilities of being in regime two (high unconditional volatility regime), $\mathsf{P}[S_t = 2|\mathcal{I}_T]$ for $t = 1, \ldots, T$, superimposed on the SMI log-returns (top graph) as well as the filtered volatility of the overall process (bottom graph). Filtered (annualized) volatilities are extracted from estimated object using the function `Volatility()`:

```
R> vol <- sqrt(250) * Volatility(fit.ml)
```

We refer the reader to `help("Volatility")` for further details. As expected, when the smoothed probabilities of regime two are near one, the filtered volatility of the process sharply increases. Interestingly, we further note that the Markov chain evolves persistently over time and that, in the limit, as reported by the `summary()` (`Stable probabilities`), the probabilities of being in the two states are about 54% and 46%. As documented by Ardia (2008) and Mullen *et al.* (2011), ML estimation can be difficult for MSGARCH-type models. Fortunately, MCMC procedures can be used to explore the joint posterior distribution of the model parameters, thus avoiding convergence to local maxima commonly encountered via ML estimation. The Bayesian approach offers additional advantages. For instance, exact

distributions of nonlinear functions of the model parameters can be obtained at low cost by simulating from the joint posterior distribution, and parameter uncertainty can be integrated in the forecasts through the predictive distribution.

To estimate MSGARCH-type models from a Bayesian viewpoint, the R package **MSGARCH** relies on the adaptive MCMC sampler of Vihola (2012).[16] Specifically, the estimation method is a random-walk Metropolis-Hastings algorithm with coerced acceptance rate. We observed excellent performance in the context of (identified) mixture models. Using the ML parameter estimates as starting values, we can estimate the model by MCMC as follows:

```
R> nmcmc <- 12500
R> nburn <- 5000
R> nthin <- 5
R> ctr <- list(nmcmc = nmcmc, nburn = nburn, nthin = nthin,
+    par0 = fit.ml$par)
R> fit.mcmc <- FitMCMC(ms2.gjr.s, data = SMI, ctr = ctr)
R> summary(fit.mcmc)


Specification type: Markov-switching
Specification name: gjrGARCH_std gjrGARCH_std
Number of parameters in each variance model: 4 4
Number of parameters in each distribution: 1 1
-----------------------------------------
Fixed parameters:
None
-----------------------------------------
Across regime constrained parameters:
nu
-----------------------------------------
Posterior sample (size: 2500)
           Mean     SD     SE   TSSE    RNE
alpha0_1 0.2101 0.0323 0.0006 0.0015 0.1826
alpha1_1 0.0007 0.0002 0.0000 0.0000 0.2079
alpha2_1 0.2110 0.0408 0.0008 0.0017 0.2251
beta_1   0.5216 0.0526 0.0011 0.0026 0.1668
nu_1     9.6161 1.4718 0.0294 0.0753 0.1528
alpha0_2 0.1417 0.0454 0.0009 0.0021 0.1803
alpha1_2 0.0062 0.0024 0.0000 0.0001 0.1839
alpha2_2 0.1839 0.0425 0.0009 0.0021 0.1671
beta_2   0.8316 0.0359 0.0007 0.0017 0.1749
P_1_1    0.9971 0.0013 0.0000 0.0001 0.2216
P_2_1    0.0045 0.0017 0.0000 0.0001 0.1765
-----------------------------------------
Posterior mean transition matrix:
      t+1|k=1 t+1|k=2
t|k=1  0.9971  0.0029
```

[16] However, as reported in Section 3.2, the user is free to implement his own sampler.

```
t|k=2  0.0045  0.9955
----------------------------------------
Posterior mean stable probabilities:
State 1 State 2
 0.6137  0.3863
----------------------------------------
Acceptance rate MCMC sampler: 28.3%
nmcmc: 12500
nburn: 5000
nthin: 5
----------------------------------------
DIC: 6721.4909
----------------------------------------
```

The posterior distribution of mixture and Markov-switching models often exhibits non-elliptical shapes which lead to non-reliable estimation of the uncertainty of model parameters (see, e.g., Ardia, Hoogerheide, and Van Dijk 2009). This invalidates the use of the Gaussian asymptotic distribution for inferential purposes in finite samples. Our results display this characteristic as shown in Figure 3 where we plot 2,500 draws of the posterior sample for the parameters $\alpha_{1,1}$ and $\alpha_{1,2}$. The blue square reports the posterior mean while the red triangle reports the ML estimate. An interesting aspect of the Bayesian estimation is that we can make distributional (probabilistic) statements on any (possibly nonlinear) function of the model parameters. This is achieved by simulation. For instance, for each draw in the posterior sample we can compute the unconditional volatility in each regime, to get its posterior distribution. Figure 4 displays the posterior distributions of the unconditional annualized volatility in each regime. In the low-volatility regime, the distribution is centered around 8.9% per annum. For the high-volatility regime, the distribution is centered around 32.7% per annum. The 95% confidence bands given by the Bayesian approach are [7.9%, 10.3%] and [25.0%, 46.6%], respectively. Notice that both distributions exhibit positive skewness. Hence, relying on the asymptotic normal approximation with the delta method would yield erroneous estimates of the 95% confidence band of the unconditional volatility in each regime. We show now how to include parameter uncertainty in the one-step ahead predictive density of MSGARCH models. The predictive density is computed via the `PredPdf()` function defined for '`MSGARCH_ML_FIT`' and '`MSGARCH_MCMC_FIT`' objects. The arguments of the `PredPdf()` function are:

- `object`: An object of class '`MSGARCH_ML_FIT`', '`MSGARCH_MCMC_FIT`', or '`MSGARCH_SPEC`'.
- `x`: A numeric vector of mesh points on the domain for which the predictive distribution is calculated.
- `par`: A numeric vector of model parameters if `object` is of class '`MSGARCH_SPEC`'. By default `par = NULL`.
- `data`: A numeric vector of observations if `object` is of class '`MSGARCH_SPEC`'. By default `data = NULL`.
- `log`: A logical indicating if the logarithm of the density is returned. By default, `log = FALSE`.

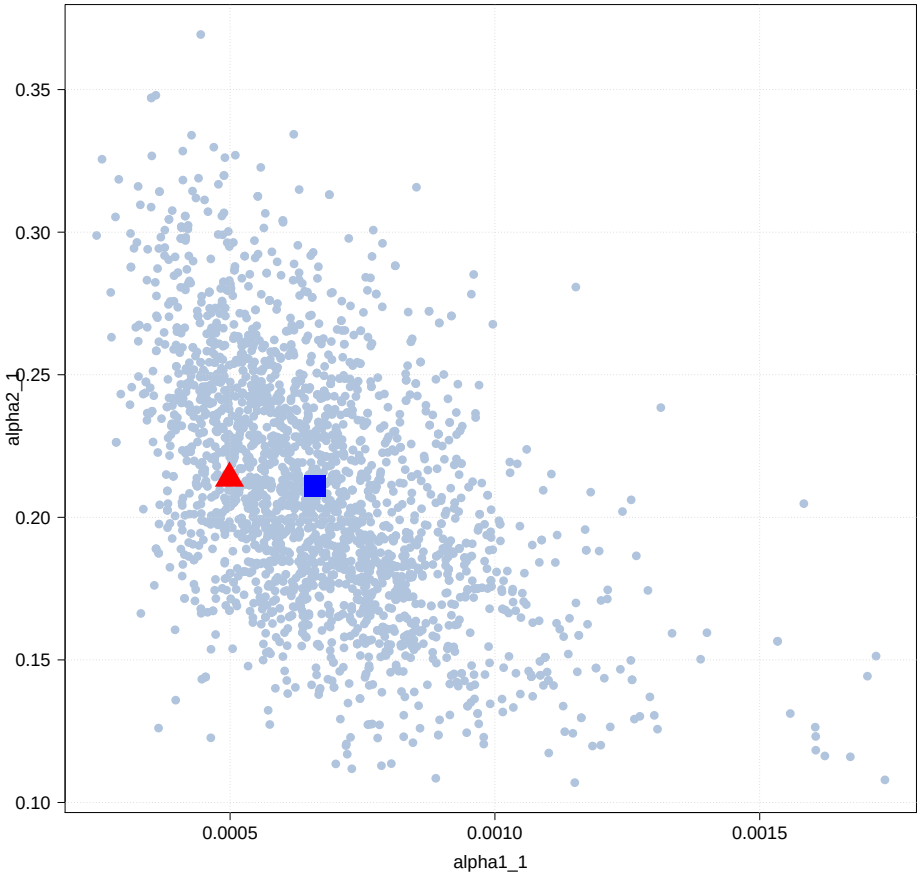

Figure 3: Scatter plot of posterior draws from the marginal distribution of $(\alpha_{1,1}, \alpha_{1,2})^\top$ obtained with the adaptive random walk strategy. The blue square reports the posterior mean, and the red triangle reports the ML estimate. The graph is based on 2,500 draws from the joint posterior sample.

- `do.cumulative`: A logical indicating if the predictive density should be computed on the distribution of simulated cumulative values. This is useful in the case of log-returns, to compute the predictive density of multi-period returns. By default, `do.cumulative = FALSE`.

Additional arguments are: `nahead`, `do.its`, and `ctr`; see `help("PredPdf")`. As for other functions available in the R package **MSGARCH**, when an object of class '`MSGARCH_MCMC_FIT`' is used, the output is computed accounting for parameter uncertainty. For instance, if we want to evaluate the one-step ahead predictive density in the range of values from -5 to 0, we run:

```
R> nmesh <- 1000
R> x <- seq(from = -5, to = 0, length.out = nmesh)
R> pred.mle <- as.vector(PredPdf(fit.ml, x = x, nahead = 1))
R> pred.bay <- as.vector(PredPdf(fit.mcmc, x = x, nahead = 1))
```

where `pred.ml` and `pred.mcmc` are numeric vectors of length 1,000. We stress that `PredPdf()`

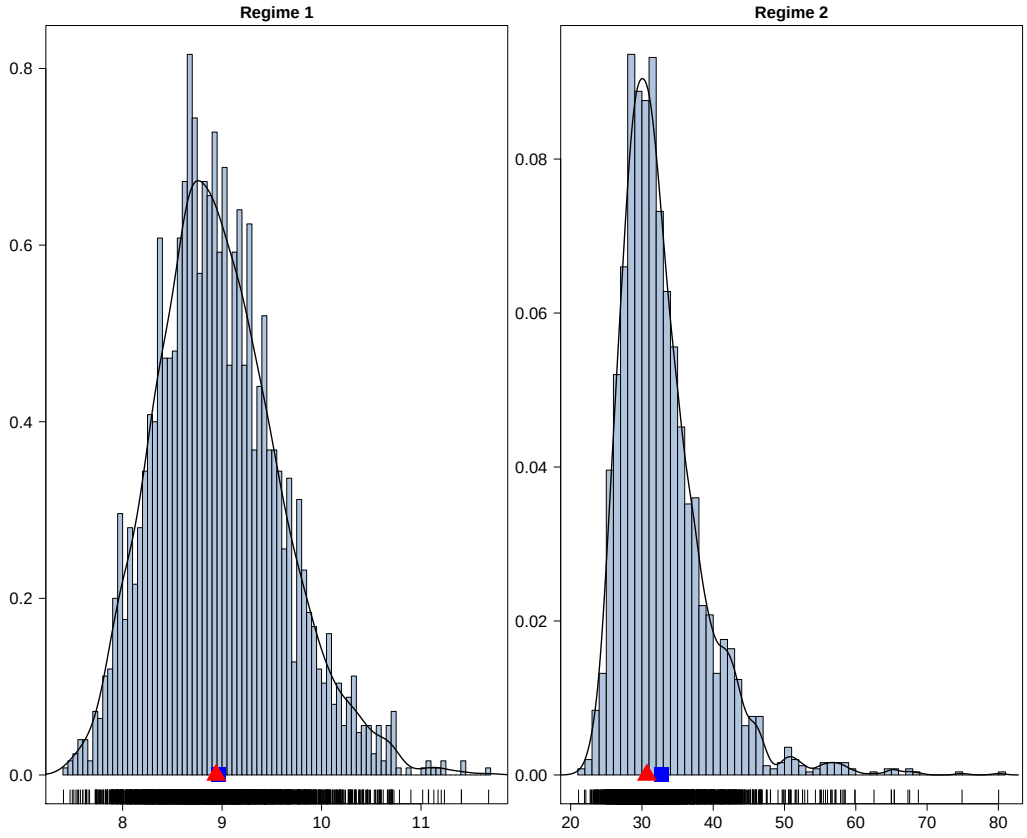

Figure 4: Histograms of the posterior distribution for the unconditional volatility in each regime. Both graphs are based on 2,500 draws from the joint posterior sample. The blue square reports the posterior mean, and the red triangle reports the ML estimate.

is also defined for objects of class 'MSGARCH_SPEC' such that, after providing the vector of parameters `par`, and a series of observations `data`, the predictive distribution can be evaluated also for non-estimated models.[17] Another possibility is to extract the posterior draws from an estimated object of class 'MSGARCH_MCMC_FIT' and to compute the predictive density for each of them. This way, we can investigate the impact of the parameter uncertainty on the conditional distribution as, for example, in Ardia, Kolly, and Trottier (2017). This is performed with the following commands:

```
R> draws <- fit.mcmc$par
R> pred.draws <- matrix(data = NA, nrow = nrow(draws), ncol = nmesh)
R> for (i in 1:nrow(draws)) {
+    tmp <- PredPdf(ms2.gjr.s, par = draws[i, ], x = x, data = SMI,
+      nahead = 1)
+    pred.draws[i, ] <- as.vector(tmp)
+  }
```

[17] This is also the case for the functions `Risk()`, `PIT()`, `Volatility()` and `predict()`; see the related R documentation.

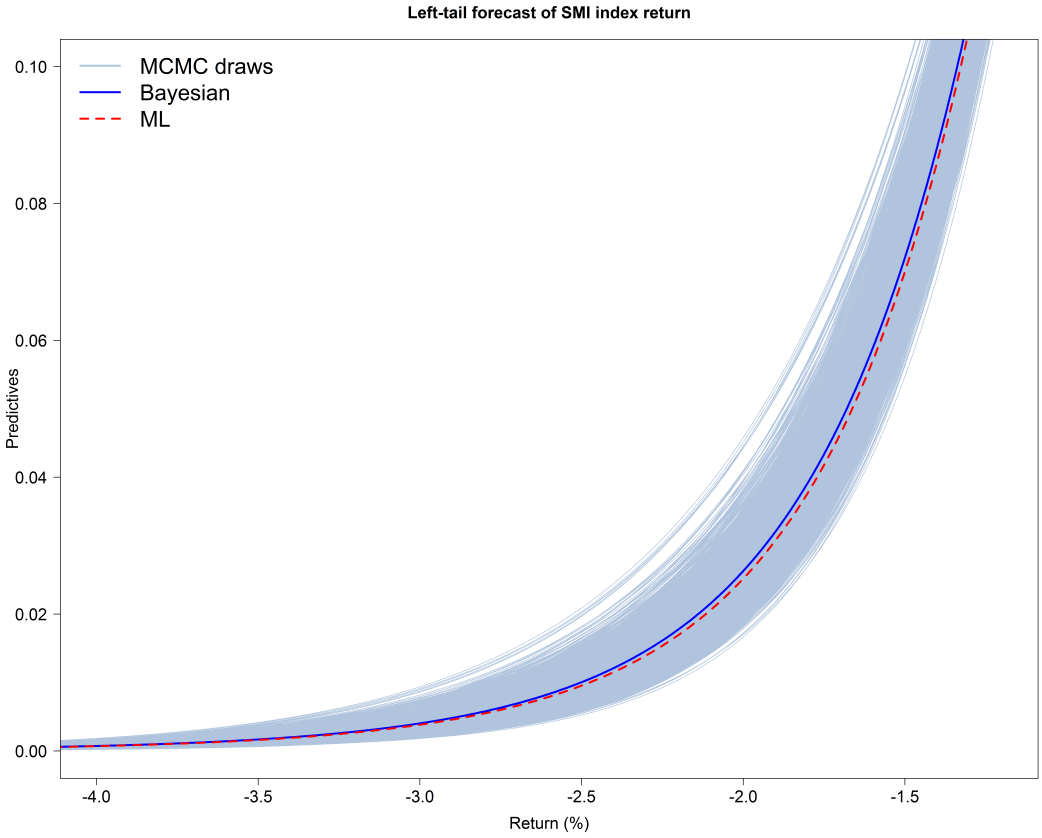

Figure 5: Left tail of the one-step ahead predictive distribution for the SMI log-returns. The solid blue line reports the predictive Bayesian density (integrating the parameter uncertainty), the dashed red line reports the ML conditional density, and the solid light-blue lines the conditional densities obtained for each of the 2,500 draws in the posterior sample.

In Figure 5, we display the left tail of the one-step ahead distribution for the log-returns. We see why integrating parameter uncertainty can be useful for left-tail prediction. The Bayesian predictive density (solid blue line) is a particular average of the predictive densities that can be formed with individual posterior MCMC draws (thin solid blue lines). It is generally more conservative than the predictive density with plugged ML estimates (dashed red line) and offers additional flexibility by accounting for all likely scenarios within the model structure.

As a last illustration in our empirical section, we provide the code for a backtest analysis of the `ms2.gjr.s` model against its single-regime counterpart. First we create the single-regime model and gather both specifications in a list.

```
R> gjr.s <- CreateSpec(variance.spec = list(model = "gjrGARCH"),
+    distribution.spec = list(distribution = "std"),
+    switch.spec = list(K = 1))
R> models <- list(gjr.s, ms2.gjr.s)
```

Then we define the backtest settings. We decide to test the performance of the models over 1,000 out-of-sample observations and focus on the one-step ahead value-at-risk forecasts at

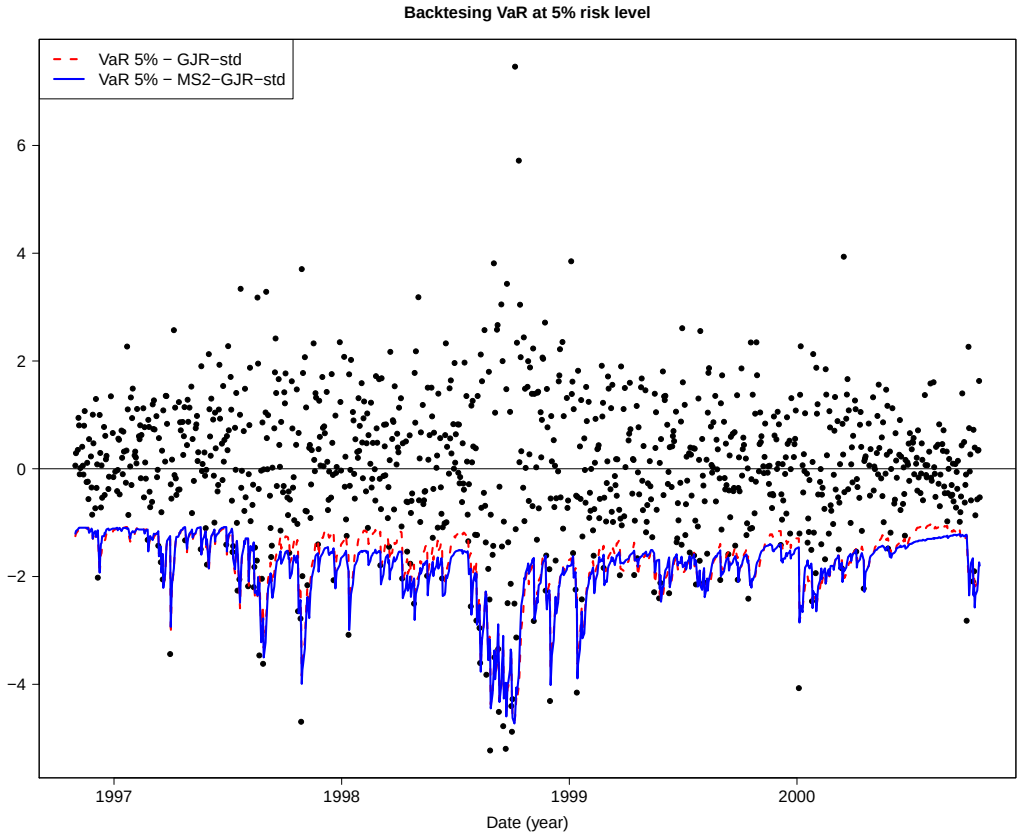

Figure 6: One-day ahead VaR forecasts at the 5% risk level provided by the Markov-switching (solid blue line) and single-regime (dashed red line) specifications together with the realized returns (black dots).

the 5% risk level. Forecasts are based on rolling windows of 1,500 observations, and models are re-estimated every 100 observations for speed of illustration purposes.

```
R> n.ots <- 1000
R> n.its <- 1500
R> alpha <- 0.05
R> k.update <- 100
```

We initialize the vector of out-of-sample returns and the matrix of VaR forecasts. Then, we loop over the observations in the out-of-sample window. For each new observation, we use the last 1,500 data to predict the one-step ahead VaR at the 5% level, and store it. Models are estimated by ML every 100 observations in the out-of-sample window.

```
R> VaR <- matrix(NA, nrow = n.ots, ncol = length(models))
R> y.ots <- matrix(NA, nrow = n.ots, ncol = 1)
R> model.fit <- vector(mode = "list", length = length(models))
R> for (i in 1:n.ots) {
+    cat("Backtest - Iteration: ", i, "\n")
+    y.its <- SMI[i:(n.its + i - 1)]
```

```
+    y.ots[i] <- SMI[n.its + i]
+    for (j in 1:length(models)) {
+      if (k.update == 1 || i %% k.update == 1) {
+        cat("Model", j, "is reestimated\n")
+        model.fit[[j]] <- FitML(spec = models[[j]], data = y.its,
+          ctr = list(do.se = FALSE))
+      }
+      VaR[i, j] <- Risk(model.fit[[j]]$spec, par = model.fit[[j]]$par,
+        data = y.its, n.ahead = 1, alpha = alpha, do.es = FALSE,
+        do.its = FALSE)$VaR
+    }
+  }
```

In Figure 6 we display the resulting VaR forecasts of the two models together with the realized returns. We notice the discrepancy between the forecasts in 1998. The economically relevant question is then to evaluate which of the VaR forecasts are most accurate in terms of correctly predicting the α-quantile loss such that we expect to have a proportion α of exceedances. We use the R package **GAS** (Ardia *et al.* 2019b) to compute the p values of two backtesting hypothesis tests of correct conditional coverage of the VaR: the conditional coverage (CC) test by Christoffersen (1998) and the dynamic quantile (DQ) test by Engle and Manganelli (2004). Both tests aim at determining if the VaR forecasts achieve correct unconditional coverage and if the violations of the VaR are independent over time. Both should be fulfilled under correct model specification.

```
R> library("GAS")
R> CC.pval <- DQ.pval <- vector("double", length(models))
R> for (j in 1:length(models)) {
+    test <- GAS::BacktestVaR(data = y.ots, VaR = VaR[, j],
+      alpha = alpha)
+    CC.pval[j] <- test$LRcc[2]
+    DQ.pval[j] <- test$DQ$pvalue
+  }
R> names(CC.pval) <- names(DQ.pval) <- c("GJR-std", "MS2-GJR-std")
R> print(CC.pval)
```

For both tests, we notice the best performance of the Markov-switching specification.

```
GJR-std MS2-GJR-std
0.02092     0.08402
```

```
R> print(DQ.pval)
```

```
GJR-std MS2-GJR-std
0.03478     0.14137
```

5. Conclusion

We introduced the R package **MSGARCH** which allows users to estimate, simulate, and perform forecasts with Markov-switching GARCH models in the R statistical software. We detailed how to create various single-regime and regime-switching specifications with different scedastic functions and conditional distributions. We documented how to perform maximum likelihood and Bayesian estimations of these models. In an empirical illustration with financial returns, we showed how to estimate, select, and forecast with this class of models.

The R language has become an important vehicle for knowledge transfer in time series analysis over the last years. We hope the R package **MSGARCH** will be useful for academics and practitioners to improve their conditional volatility and density forecasts in the broad range of applications involving regime changes in volatility dynamics.

Finally, if you use the R package **MSGARCH**, please cite the software in publications using `citation(package = "MSGARCH")`.

Computational details

The results in this paper were obtained using R 3.5.0 (R Core Team 2018) with the packages **coda** version 0.19-1 (Plummer *et al.* 2006), **GAS** version 0.2.6 (Ardia *et al.* 2019b), **numDeriv** version 2016.8-1 (Gilbert and Varadhan 2016), **MSGARCH** version 2.3 (Ardia *et al.* 2019a), **Rcpp** version 0.12.16 (Eddelbuettel *et al.* 2018; Eddelbuettel and François 2011), **RcppArmadillo** version 0.8.500.0 (Eddelbuettel *et al.* 2017; Eddelbuettel and Sanderson 2014), and **zoo** version 1.8-1 (Zeileis and Grothendieck 2005). Computations were performed on Windows 10 x86 64-w64-mingw32/x64 (64-bit) with Intel(R) Xeon(R) CPU E5-2650 2x 2.30 GHz. Code for the computations is available in the R script `v91i04.R`, available in the supplementary materials and also in the `Examples` folder of the GitHub repository at https://github.com/keblu/MSGARCH.

The results in this paper rely on simulations from an adaptive MCMC scheme. Therefore, results depend on the `set.seed` value and on the linear algebra library used, as the adaptive scheme requires Cholesky decomposition and eigenvalue calculation. While this would have a negligible impact for a single draw, the difference accumulates over the MCMC iterations, and can lead to different MCMC posterior results than the ones reported in this paper. An upper bound on the magnitude of those differences is difficult to evaluate. In the comparison between Windows 10 and Ubuntu 18.04.3, as reported in the supplementary material and in the folder `Examples` on the GitHub repository, we find differences of up to the third digit for the MCMC results of Section 3.2 and up to the second digit for the MCMC results of Section 4.

In the script `v91i04-timing.R`, we provide a timing comparison between the ML and MCMC estimation strategies. The estimation of a two-regime MSGARCH model with normal innovations on the `SMI` dataset (2,500 observations) is around ten times faster with the ML approach than with the MCMC approach (2.8 seconds instead of 26.7 seconds). We also perform a comparison between the package **MSGARCH** and the R package **bayesGARCH** (Ardia and Hoogerheide 2010) in the case of a single-regime GARCH$(1, 1)$ model with normal innovations. We observe that **MSGARCH** is not only competitive in terms of computational time but also for accuracy (measured with the effective number of draws). With the default settings (5,000 burn-in draws, 10,000 MCMC draws and a thinning of 10, thus a posterior

sample of size 1,000), the estimation with **MSGARCH** is about ten times faster (6.5 against 69.1 seconds) and the effective number of draws is 3.1 times larger.

R (R Core Team 2018) itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/. The **MSGARCH** version under development is available on the website at http://keblu.github.io/MSGARCH/.

Acknowledgments

The authors are grateful to the Associate Editor (Rob J. Hyndman) and two anonymous referees for useful comments. They also thank Carol Alexander, Samuel Borms, Muriel Buri, Peter Carl, Dirk Eddelbuettel, Laurent Fastnacht, Félix-Antoine Fortin, Alexios Ghalanos, Richard Gerlach, Lennart Hoogerheide, Eliane Maalouf, Brian Peterson, Enrico Schumann, Tobias Setz, Max Tchirikov, conference participants at R/Finance 2017 (Chicago), the 37th International Symposium on Forecasting (Cairns), useR! 2017 (Brussels), Quant Insights 2017 (London), MAFE 2018 (Madrid), eRum 2018 (Budapest), and seminar participants at HEC Liège, Paris-Dauphine, and IAE-AMSE Aix-Marseille. We acknowledge Industrielle-Alliance, International Institute of Forecasters (https://forecasters.org), Google Summer of Code (https://summerofcode.withgoogle.com), FQRSC (http://www.frqsc.gouv.qc.ca, grant 2015-NP-179931), swissuniversities (https://www.swissuniversities.ch), IVADO (https://ivado.ca), and the Swiss National Science Foundation (http://www.snf.ch, grant 179281) for their financial support, and Calcul Québec (https://www.calculquebec.ca) for computational support.

References

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

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 Heidelberg. doi:10.1007/978-3-540-78657-3.

Ardia D, Bluteau K, Boudt K, Catania L (2018). "Forecasting Risk with Markov-Switching GARCH Models: A Large-Scale Performance Study." *International Journal of Forecasting*, **34**(4), 733–747. doi:10.1016/j.ijforecast.2018.05.004.

Ardia D, Bluteau K, Boudt K, Catania L, Ghalanos A, Peterson B, Trottier DA (2019a). ***MSGARCH**: Markov-Switching GARCH Models in R*. R package version 2.31, URL https://CRAN.R-project.org/package=MSGARCH.

Ardia D, Boudt K, Catania L (2019b). "Generalized Autoregressive Score Models in R: The **GAS** Package." *Journal of Statistical Software*, **88**(6), 1–28. doi:10.18637/jss.v088.i06.

Ardia D, Hoogerheide LF (2010). "Bayesian Estimation of the GARCH(1, 1) Model with Student-*t* Innovations." *The R Journal*, **2**(2), 41–47. doi:10.32614/rj-2010-014.

Ardia D, Hoogerheide LF, Van Dijk HK (2009). "Adaptive Mixture of Student-t Distributions as a Flexible Candidate Distribution for Efficient Simulation: The R Package **AdMit**." *Journal of Statistical Software*, **29**(3), 1–32. doi:10.18637/jss.v029.i03.

Ardia D, Kolly J, Trottier DA (2017). "The Impact of Parameter and Model Uncertainty on Market Risk Predictions from GARCH-Type Models." *Journal of Forecasting*, **36**(7), 808–823. doi:10.1002/for.2472.

Bauwens L, Backer B, Dufays A (2014). "A Bayesian Method of Change-Point Estimation with Recurrent Regimes: Application to GARCH Models." *Journal of Empirical Finance*, **29**, 207–229. doi:10.1016/j.jempfin.2014.06.008.

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.

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

Brooks C, Burke SP, Persand G (2001). "Benchmarks and the Accuracy of GARCH Model Estimation." *International Journal of Forecasting*, **17**(1), 45–56. doi:10.1016/s0169-2070(00)00070-4.

Cai J (1994). "A Markov Model of Switching-Regime ARCH." *Journal of Business & Economic Statistics*, **12**(3), 309–316. doi:10.2307/1392087.

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.

Christoffersen PF (1998). "Evaluating Interval Forecasts." *International Economic Review*, **39**(4), 841–862. doi:10.2307/2527341.

Dueker MJ (1997). "Markov Switching in GARCH Processes and Mean-Reverting Stock-Market Volatility." *Journal of Business & Economic Statistics*, **15**(1), 26–34. doi:10.2307/1392070.

Eddelbuettel D (2019). *CRAN Task View: Empirical Finance*. Version 2019-03-07, URL https://CRAN.R-project.org/view=Finance.

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 JJ, Ushey K, Kou Q, Bates D, Chambers J (2018). ***Rcpp**: Seamless R and C++ Integration*. R package version 0.12.16, URL https://CRAN.R-project.org/package=Rcpp.

Eddelbuettel D, François R, Bates D (2017). ***RcppArmadillo**: **Rcpp** Integration for the **Armadillo** Templated Linear Algebra Library*. R package version 0.8.500.0, URL https://CRAN.R-project.org/package=RcppArmadillo.

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

Engle RF (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." *Econometrica*, **50**(4), 987–1007. doi:10.2307/1912773.

Engle RF, Manganelli S (2004). "CAViaR: Conditional Autoregressive Value at Risk by Regression Quantiles." *Journal of Business & Economic Statistics*, **22**(4), 367–381. doi:10.1198/073500104000000370.

Fernández C, Steel MFJ (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.

Frühwirth-Schnatter S (2006). *Finite Mixture and Markov Switching Models*. 1st edition. Springer-Verlag, New York. doi:10.1007/978-0-387-35768-3.

Geweke J (2007). "Interpretation and Inference in Mixture Models: Simple MCMC Works." *Computational Statistics & Data Analysis*, **51**(7), 3529–3550. doi:10.1016/j.csda.2006.11.026.

Ghalanos A (2017). ***rugarch***: *Univariate GARCH Models*. R package version 1.3-8, URL https://CRAN.R-project.org/package=rugarch.

Gilbert P, Varadhan R (2016). ***numDeriv***: *Accurate Numerical Derivatives*. R package version 2016.8-1, URL https://CRAN.R-project.org/package=numDeriv.

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

Gray SF (1996). "Modeling the Conditional Distribution of Interest Rates as a Regime-Switching Process." *Journal of Financial Economics*, **42**(1), 27–62. doi:10.1016/0304-405x(96)00875-6.

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*, **57**(2), 357–384. doi:10.2307/1912559.

Hamilton JD (1994). *Time Series Analysis*. Princeton University Press, Princeton.

Hamilton JD, Susmel R (1994). "Autoregressive Conditional Heteroskedasticity and Changes in Regime." *Journal of Econometrics*, **64**(1–2), 307–333. doi:10.1016/0304-4076(94)90067-1.

Hoogerheide L, Van Dijk HK (2010). "Bayesian Forecasting of Value at Risk and Expected Shortfall Using Adaptive Importance Sampling." *International Journal of Forecasting*, **26**(2), 231–247. doi:10.1016/j.ijforecast.2010.01.007.

Hyndman RJ (2019). *CRAN Task View: Time Series Analysis*. Version 2019-08-25, URL https://CRAN.R-project.org/view=TimeSeries.

Kastner G (2016). "Dealing with Stochastic Volatility in Time Series Using the R Package **stochvol**." *Journal of Statistical Software*, **69**(5), 1–30. doi:10.18637/jss.v069.i05.

Kim CJ, Nelson CR (1999). "Has the U.S. Economy Become More Stable? A Bayesian Approach Based on a Markov-Switching Model of the Business Cycle." *Review of Economics and Statistics*, **81**(4), 608–616. doi:10.1162/003465399558472.

Klaassen F (2002). "Improving GARCH Volatility Forecasts with Regime-Switching GARCH." In *Advances in Markov-Switching Models*, pp. 223–254. Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-642-51182-0_10.

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

McNeil AJ, Frey R, Embrechts P (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. 2nd edition. Princeton University Press.

Mullen KM, Ardia D, Gil DL, Windover D, Cline J (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*, **59**(2), 347–370. doi:10.2307/2938260.

Nobel Media (2003). "The Prize in Economic Sciences 2003 – Press Release." URL https://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2003/press.html.

Papastamoulis P (2016). "**label.switching**: An R Package for Dealing with the Label Switching Problem in MCMC Outputs." *Journal of Statistical Software*, **69**(1), 1–24. doi:10.18637/jss.v069.c01.

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://www.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Pretis F, Reade JJ, Sucarrat G (2018). *Automated General-to-Specific (GETS) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks*. doi:10.18637/jss.v086.i03.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. R version 3.5.0, URL https://www.R-project.org/.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464.

Sousa T, Otiniano C, Lopes S, Diethelm W (2015). ***GEVStableGarch**: ARMA-GARCH/APARCH Models with GEV and Stable Distributions*. R package version 1.1, URL https://CRAN.R-project.org/package=GEVStableGarch.

Spiegelhalter DJ, Best NG, Carlin BP, Van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B*, **64**(4), 583–639. doi:10.1111/1467-9868.00353.

Sucarrat G (2015). ***lgarch**: Simulation and Estimation of Log-GARCH Models*. R package version 0.6-2, URL https://CRAN.R-project.org/package=lgarch.

Teräsvirta T (2009). "An Introduction to Univariate GARCH Models." In *Handbook of Financial Time Series*, pp. 17–42. Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-540-71297-8_1.

Trapletti A, Hornik K (2017). ***tseries**: Time Series Analysis and Computational Finance*. R package version 0.10-42, URL https://CRAN.R-project.org/package=tseries.

Trottier DA, Ardia D (2016). "Moments of Standardized Fernandez-Steel Skewed Distributions: Applications to the Estimation of GARCH-Type Models." *Finance Research Letters*, **18**, 311–316. doi:10.1016/j.frl.2016.05.006.

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.

Viterbi A (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." *IEEE Transactions on Information Theory*, **13**(2), 260–269. doi:10.1109/tit.1967.1054010.

Wuertz D, Chalabi Y, Miklovic M, Boudt K, Chausse P (2016). ***fGarch**: Rmetrics – Autoregressive Conditional Heteroskedastic Modelling*. R package version 3010.82.1, URL https://CRAN.R-project.org/package=fGarch.

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

Zeileis A, Grothendieck G (2005). "**zoo**: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software*, **14**(6), 1–27. doi:10.18637/jss.v014.i06.

Zhang Y, Wang J, Wang X (2014). "Review on Probabilistic Forecasting of Wind Power Generation." *Renewable and Sustainable Energy Reviews*, **32**, 255–270. doi:10.1016/j.rser.2014.01.033.

Affiliation:

David Ardia *(corresponding author)*
Department of Decision Sciences
HEC Montréal
3000, Ch. de la Côte-Sainte-Catherine, Montréal (Québec) H3T 2A7, Canada
E-mail: david.ardia@hec.ca

Keven Bluteau
Institute of Financial Analysis
University of Neuchâtel
Rue A.-L. Breguet 2, 2000 Neuchâtel, Switzerland
and
Solvay Business School
Vrije Universiteit Brussel, Belgium
E-mail: keven.bluteau@unine.ch

Kris Boudt
Department of Economics
Ghent University
Sint-Pietersplein 5, 9000 Gent, Belgium
and
Solvay Business School
Vrije Universiteit Brussel, Belgium *and*
Econometrics and Finance Department
Vrije Universiteit Amsterdam, The Netherlands
E-mail: kris.boudt@vub.be

Leopoldo Catania
Department of Economics and Business Economics
Aarhus University & CREATES
Fuglesangs Allé 4, 8210, Aarhus V, Denmark
E-mail: leopoldo.catania@econ.au.dk

Denis-Alexandre Trottier
Faculty of Business Administration
Laval University
Pavillon Palasis-Prince
2325, rue de la Terrasse, Québec, G1V 0A6, Canada
E-mail: denis-alexandre.trottier.1@ulaval.ca

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2019, Volume 91, Issue 4 *Submitted:* 2017-10-30
doi:10.18637/jss.v091.i04 *Accepted:* 2018-06-23