Generalized Autoregressive Score Models in R: The GAS Package

David Ardia
University of Neuchâtel
Laval University

Kris Boudt
Vrije Universiteit Brussel
Vrije Universiteit Amsterdam

Leopoldo Catania
University of Rome
"Tor Vergata"

Abstract

This paper presents the R package **GAS** for the analysis of time series under the Generalized Autoregressive Score (GAS) framework of Creal *et al.* (2013) and Harvey (2013). The distinctive feature of the GAS approach is the use of the score function as the driver of time–variation in the parameters of nonlinear models. The **GAS** package provides functions to simulate univariate and multivariate GAS processes, estimate the GAS parameters and to make time series forecasts. We illustrate the use of the **GAS** package with a detailed case study on estimating the time–varying conditional densities of financial asset returns.

Keywords: GAS, time series models, score models, dynamic conditional score, R software.

1. Introduction

Time–variation in the parameters describing a stochastic time series process is pervasive in almost all applied scientific fields. Early references to time series models include Kalman (1960) and Box and Jenkins (1970). In many settings, the model of interest is characterized by time–varying parameters, for which the literature has proposed a myriad of possible specifications. Recently, Creal *et al.* (2013) and Harvey (2013) note that many of the proposed models are either difficult to estimate (in particular, the class of stochastic volatility models reviewed in Shephard (2005)) and/or do not properly take the shape of the conditional distribution of the data into account.[1] Creal *et al.* (2013) and Harvey (2013) therefore propose to use the score of the conditional density function as the main driver of time–variation in the parameters of the time series process used to describe the data. A further advantage of using the conditional score as driver is that the estimation by Maximum Likelihood is straightforward. The resulting model is referred to as: Score–Driven model, Dynamic Conditional Score (DCS) model, or Generalized Autoregressive Score (GAS) model. In this article and accompanying R package, we use the GAS acronym.

The R package **GAS** is conceived to be of relevance for the modelling of all types of time series data. It does not matter whether they are real–valued, integer–valued, (0,1)–bounded or strictly positive, as long as there is a conditional density for which the score function and the Hessian are well–defined. The practical relevance of the GAS framework has been illustrated in the case of financial risk forecasting (see *e.g.*, Harvey and Sucarrat (2014) for market risk, Oh and Patton (2016) for systematic risk, and Creal *et al.* (2014) for credit risk analysis), dependence modelling (see *e.g.*, Harvey and Thiele (2016) and Janus *et al.* (2014)), and spatial econometrics (see *e.g.*, Blasques *et al.* (2016b) and Catania and Billé (2017)). For a more complete overview of the work on GAS models, we refer the reader to the GAS community page at http://www.gasmodel.com/.

It is important to note that, even though the GAS framework has been developed by econometricians, it is flexible enough to be used in all fields in which the use of time–varying parameter models is relevant. The main difficulty in using GAS models is to derive the score and Hessian and implementing the Maximum Likelihood estimation of the resulting nonlinear models. The R package **GAS** answers these needs by proposing an integrated set of R functions to do time series analysis in the R statistical language (R Core Team 2017) under the GAS framework. The functionalities include: (i) estimation, (ii) prediction, (iii) simulation, (iv) backtesting, and (v) graphical representation of the results, implying that it is ready to use in real–life applications. The user interface uses the R programming language, which has the advantage of being free and open source. However, most of the underlying routines are principally written in C++ exploiting the **armadillo** library (Sanderson 2010) and the R packages **Rcpp** (Eddelbuettel and François 2011; Eddelbuettel *et al.* 2017a) and **RcppArmadillo** (Eddelbuettel and Sanderson 2014; Eddelbuettel *et al.* 2017b) to speed up the computations. Furthermore, the package is written using S4 classes, and provides methods such as `coef()`, `plot()` and `show()` to extract and analyze the results. We believe

[1] A typical example is the class of (G)ARCH models in which the squared (demeaned) return is the driver of time–variation in the conditional variance, independently of the shape of the conditional distribution of the return. To see that this is counter–intuitive, consider the case of observing a 10% return when the conditional mean is 0% and the volatility is 3%. Under the assumption of a Gaussian distribution, the 10% return is a strong signal of an increase in volatility, while under a fat–tailed Student–t distribution, the signal is weakened because of the higher probability that the extreme value is an observation from the tails.

that this functionality makes the R package **GAS** more user friendly and thus accessible to a larger number of potential users. The R package **GAS** is available from the CRAN repository at https://cran.r-project.org/package=GAS. Other codes for specific GAS models are available on the GAS community page at http://www.gasmodel.com/. For instance, the R package **betategarch** (Sucarrat 2013) has functions to estimate the beta–t–EGARCH model of Harvey (2013) and its skewed version introduced by Harvey and Sucarrat (2014).

The outline of the paper is as follows. Section 2 reviews the GAS framework to define time–varying parameter models, referring to the seminal works of Creal *et al.* (2013) and Harvey (2013). Section 3 introduces the R package **GAS** and illustrates how to simulate, estimate and make predictions. Section 4 presents a real–life application to financial return data. Section 5 concludes.

2. The GAS framework to modeling time–varying parameters

One of the most appealing characteristics of the GAS framework is its applicability to define time–varying parameter models in a large variety of univariate and multivariate time series settings. We try to be as general as possible in reviewing the GAS framework, and report in Appendix A the detailed equations for the specific case of a conditionally Student–t distributed random variable with time–varying location, scale and degrees of freedom parameters. In this section, we first introduce the notation and present the GAS model when the parameter space is unrestricted. We then show how a mapping function can be used to model the time–variation in the parameters when the parameter space is restricted. The section concludes by summarizing the Maximum Likelihood approach for GAS model estimation.

2.1. Model specification

Let $\mathbf{y}_t \in \Re^N$ be an N–dimensional random vector at time t with conditional distribution:

$$\mathbf{y}_t | \mathbf{y}_{1:t-1} \sim p(\mathbf{y}_t; \boldsymbol{\theta}_t) \,, \tag{1}$$

where $\mathbf{y}_{1:t-1} \equiv (\mathbf{y}_1^\top, \ldots, \mathbf{y}_{t-1}^\top)^\top$ contains the past values of $\mathbf{y}_t$ up to time $t-1$ and $\boldsymbol{\theta}_t \in \Theta \subseteq \Re^J$ is a vector of time–varying parameters which fully characterizes $p(\cdot)$ and only depends on $\mathbf{y}_{1:t-1}$ and a set of static additional parameters $\boldsymbol{\xi}$, *i.e.*, $\boldsymbol{\theta}_t \equiv \boldsymbol{\theta}(\mathbf{y}_{1:t-1}, \boldsymbol{\xi})$ for all t. The main feature of GAS models is that the evolution in the time–varying parameter vector $\boldsymbol{\theta}_t$ is driven by the score of the conditional distribution defined in (1), together with an autoregressive component:

$$\boldsymbol{\theta}_{t+1} \equiv \boldsymbol{\kappa} + \mathbf{A}\, \mathbf{s}_t + \mathbf{B}\, \boldsymbol{\theta}_t \,, \tag{2}$$

where, $\boldsymbol{\kappa}$, $\mathbf{A}$ and $\mathbf{B}$ are matrices of coefficients with proper dimensions collected in $\boldsymbol{\xi}$, and $\mathbf{s}_t$ is a vector which is proportional to the score of (1):

$$\mathbf{s}_t \equiv \mathbf{S}_t(\boldsymbol{\theta}_t)\, \boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t) \,.$$

The matrix $\mathbf{S}_t$ is a $J \times J$ positive definite scaling matrix known at time t and:

$$\boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t) \equiv \frac{\partial \log p(\mathbf{y}_t; \boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t} \,,$$

is the score of (1) evaluated at $\boldsymbol{\theta}_t$. Creal *et al.* (2013) suggest to set the scaling matrix $\mathbf{S}_t$ to a power $\gamma > 0$ of the inverse of the Information Matrix of $\boldsymbol{\theta}_t$ to account for the variance of $\boldsymbol{\nabla}_t$. More precisely:

$$\mathbf{S}_t(\boldsymbol{\theta}_t) \equiv \mathcal{I}_t(\boldsymbol{\theta}_t)^{-\gamma},$$

with:

$$\mathcal{I}_t(\boldsymbol{\theta}_t) \equiv \mathsf{E}_{t-1}\left[\boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t)\boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t)^\top\right], \tag{3}$$

where the expectation is taken with respect to the conditional distribution of $\mathbf{y}_t$ given $\mathbf{y}_{1:t-1}$. The additional parameter γ is fixed by the user and usually takes value in the set $\{0, \frac{1}{2}, 1\}$. When $\gamma = 0$, $\mathbf{S}_t = \mathbf{I}$ and there is no scaling.[2] If $\gamma = 1$ ($\gamma = \frac{1}{2}$), the conditional score $\boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t)$ is premultiplied by the inverse of (the square root of) its covariance matrix $\mathcal{I}_t(\boldsymbol{\theta}_t)$.

It is worth noting that, whatever the choice of γ, $\mathbf{s}_t$ is a Martingale Difference (MD) with respect to the distribution of $\mathbf{y}_t$ given $\mathbf{y}_{1:t-1}$, *i.e.*, $\mathsf{E}_{t-1}[\mathbf{s}_t] = \mathbf{0}$ for all t. Furthermore, when $\gamma = \frac{1}{2}$, the additional moment condition $\mathsf{V}_{t-1}[\mathbf{s}_t] = \mathbf{I}$ can be easily derived. Due to the fact that $\mathbf{s}_t$ is an MD, the process $\boldsymbol{\theta}_t$ in (2) is mean–reverting around its long–term mean value $(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\kappa}$ when the spectral radius of B is less than one.[3] It follows that the J–valued vector $\boldsymbol{\kappa}$ and the $J \times J$ matrix $\mathbf{B}$ control for the level and the persistence of the process, respectively.

The additional $J \times J$ matrix of coefficients $\mathbf{A}$, that premultiplies the scaled score $\mathbf{s}_t$, controls for the impact of $\mathbf{s}_t$ on $\boldsymbol{\theta}_{t+1}$. Specifically, as detailed in Creal *et al.* (2013), the quantity $\mathbf{s}_t$ indicates the direction to update the vector of parameters from $\boldsymbol{\theta}_t$, to $\boldsymbol{\theta}_{t+1}$, acting as a steepest ascent algorithm for improving the model's local fit given the current parameter position. Interestingly, this updating procedure resembles the well–known Newton–Raphson algorithm. Hence, $\mathbf{A}$ can be interpreted as the step of the update, and needs to be designed in a way to not distort the signal coming from $\mathbf{s}_t$; see Section 2.3.

2.2. Reparameterization

In (2) the parameter vector $\boldsymbol{\theta}_t$ has a linear specification and is thus unbounded. In practice, the parameter space of $\boldsymbol{\theta}_t$ is often restricted ($\Theta \subset \Re^J$). For instance, when we model the scale parameter of a Student–t distribution, we need to ensure its positiveness. Even if this problem can be solved by imposing constraints on $\boldsymbol{\xi}$ (as is done in the GARCH model; see Bollerslev 1986), the standard solution under the GAS framework is to use a (usually nonlinear) link function $\Lambda(\cdot)$ that maps $\widetilde{\boldsymbol{\theta}}_t \in \Re^J$ into $\boldsymbol{\theta}_t$ and where $\widetilde{\boldsymbol{\theta}}_t \in \Re^J$ has the linear dynamic specification of (2).[4] Specifically, let $\Lambda : \Re^J \to \Theta$ be a twice–differentiable vector–valued mapping function such that $\Lambda(\widetilde{\boldsymbol{\theta}}_t) = \boldsymbol{\theta}_t$. The updating equation for $\boldsymbol{\theta}_t$ is then given by:

$$\begin{aligned} \boldsymbol{\theta}_t &\equiv \Lambda(\widetilde{\boldsymbol{\theta}}_t) \\ \widetilde{\boldsymbol{\theta}}_t &\equiv \boldsymbol{\kappa} + \mathbf{A}\widetilde{\mathbf{s}}_t + \mathbf{B}\widetilde{\boldsymbol{\theta}}_{t-1}, \end{aligned} \tag{4}$$

[2] We denote by $\mathbf{I}$ the identity matrix of appropriate size.

[3] The spectral radius of a $L \times L$ matrix $\mathbf{X}$ is defined as $\tau(\mathbf{X}) \equiv \max(|\tau_1|, \ldots, |\tau_L|)$, where τ_i is the i–th eigenvalue of $\mathbf{X}$, for $i = 1, \ldots, L$. In the R package **GAS**, we impose that $\tau(\mathbf{B}) < 1$.

[4] For instance, if we employ the identity mapping and a conditional Gaussian distribution for the innovations, we recover the well–known GARCH model of Bollerslev (1986). In these circumstances, usual constraints on the model coefficients to ensure positiveness of the conditional variance have to be satisfied. In the R package **GAS**, the exponential link function is employed for the time–varying scale parameters.

where $\widetilde{\mathbf{s}}_t \equiv \widetilde{\mathbf{S}}_t(\widetilde{\boldsymbol{\theta}}_t)\widetilde{\boldsymbol{\nabla}}_t(\mathbf{y}_t, \widetilde{\boldsymbol{\theta}}_t)$ and $\widetilde{\boldsymbol{\nabla}}_t(\mathbf{y}_t, \widetilde{\boldsymbol{\theta}}_t)$ represents the score of (1) with respect to $\widetilde{\boldsymbol{\theta}}_t$, and, consequently, $\widetilde{\mathbf{S}}_t(\widetilde{\boldsymbol{\theta}}_t)$ can depend on the information matrix of $\widetilde{\boldsymbol{\theta}}_t$ given by $\widetilde{\mathcal{I}}_t(\widetilde{\boldsymbol{\theta}}_t)$. Denote the Jacobian matrix of $\Lambda(\cdot)$ evaluated at $\widetilde{\boldsymbol{\theta}}_t$ as follows:

$$\mathcal{J}(\widetilde{\boldsymbol{\theta}}_t) \equiv \frac{\partial \Lambda(\widetilde{\boldsymbol{\theta}}_t)}{\partial \widetilde{\boldsymbol{\theta}}_t} \ .$$

Then, the following relations hold:

$$\begin{aligned} \widetilde{\boldsymbol{\nabla}}_t(\mathbf{y}_t, \widetilde{\boldsymbol{\theta}}_t) &= \mathcal{J}(\widetilde{\boldsymbol{\theta}}_t)^\top \boldsymbol{\nabla}_t(\mathbf{y}_t, \boldsymbol{\theta}_t) \\ \widetilde{\mathcal{I}}_t(\widetilde{\boldsymbol{\theta}}_t) &= \mathcal{J}(\widetilde{\boldsymbol{\theta}}_t)^\top \mathcal{I}_t(\boldsymbol{\theta}_t)\mathcal{J}(\widetilde{\boldsymbol{\theta}}_t) \, . \end{aligned}$$

This way, almost all the nonlinear constraints can be easily handled via the definition of a proper mapping function $\Lambda(\cdot)$ and its associated Jacobian matrix $\mathcal{J}(\cdot)$. The coefficients to be estimated are gathered into $\boldsymbol{\xi} \equiv (\boldsymbol{\kappa}, \mathbf{A}, \mathbf{B})$ and estimated by numerically maximizing the (log-)likelihood function as detailed in Section 2.3.[5] In Appendix B we discuss the choice of mapping functions for GAS models in more details.

2.3. Maximum likelihood estimation

A useful property of GAS models is that given the past information and the static parameter vector $\boldsymbol{\xi}$, the vector of time–varying parameters, $\boldsymbol{\theta}_t$, is perfectly predictable and the loglikelihood function can be easily evaluated via the prediction error decomposition. More precisely, for a sample of T realizations of $\mathbf{y}_t$, collected in $\mathbf{y}_{1:T}$, the vector of parameters $\boldsymbol{\xi}$ can be estimated by Maximum Likelihood (ML) as the solution of:

$$\widehat{\boldsymbol{\xi}} \equiv \arg\max_{\boldsymbol{\xi}} \ \mathcal{L}\left(\boldsymbol{\xi}; \mathbf{y}_{1:T}\right), \tag{5}$$

where:

$$\mathcal{L}\left(\boldsymbol{\xi}; \mathbf{y}_{1:T}\right) \equiv \log p\left(\mathbf{y}_1; \boldsymbol{\theta}_1\right) + \sum_{t=2}^{T} \log p\left(\mathbf{y}_t; \boldsymbol{\theta}_t\right) \ ,$$

with $\boldsymbol{\theta}_1 \equiv (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\kappa}$, and, for $t > 1$, $\boldsymbol{\theta}_t \equiv \boldsymbol{\theta}(\mathbf{y}_{1:t-1}, \boldsymbol{\xi})$. Note the dependence of $\boldsymbol{\theta}_t$ on $\boldsymbol{\xi}$ and $\mathbf{y}_{1:t-1}$.

There are two important caveats in the ML estimation of GAS models. The first one is that, from a theoretical perspective, ML estimation of GAS models is an on–going research topic. General results are reported by Harvey (2013), Blasques *et al.* (2014a) and Blasques *et al.* (2014b), while results for specific models have been derived by Andres (2014) and Blasques *et al.* (2016b).

The second one is that, even when the ML estimator is consistent and asymptotically Gaussian, the numerical maximization of the loglikelihood function in (5) can be challenging, because of the nonlinearities induced by $\Lambda\left(\cdot\right)$ and the way $\mathbf{y}_t$ enters the scaled score $\mathbf{s}_t$. Consequently, when the optimizer is gradient–based, good starting values need to be selected for GAS models. We refer the reader to Section 3.2 for more details.

[5] Clearly, the coefficients $\boldsymbol{\kappa}$, $\mathbf{A}$ and $\mathbf{B}$ in (4) are different from those of (2), however, for notational purposes, we continue to use the same notation.

Implementation of the models in the R package **GAS** follows the common approach in the GAS literature. First, matrices $\mathbf{A}$ and $\mathbf{B}$ are constrained to be diagonal. Second, in order to avoid an explosive pattern for $\widetilde{\boldsymbol{\theta}}_t$, the spectral radius of $\mathbf{B}$ is constrained to be less than one. Third, we use an exponential transformation in the estimation of each diagonal element of $\mathbf{A}$. This ensures its positiveness in order to not distort the signal coming from the conditional score $\mathbf{s}_t$.

3. The R package GAS

The R package **GAS** offers an integrated environment to deal with GAS models in R. Its structure is somehow similar to the R package **rugarch** (Ghalanos 2015b) for GARCH models, which is widely used by practitioners and academics. The similarities concern the steps the user has to do to perform her analysis as well as the type of functions she faces. Specifically, the first step is to specify the model, which means choosing: (i) the conditional distribution of the data, (ii) the set of parameters that have to vary over time and, (iii) the scaling mechanism for the conditional score. These steps are detailed in Section 3.1. Once the model is properly specified, the user can estimate the unknown parameters in $\boldsymbol{\xi}$ by numerical maximization of the log–likelihood function as detailed in Section 3.2. Finally, predictions according to the estimated model can be easily performed; see Section 3.3. Simulation of GAS models is presented in Section 3.4.

Functions for: (i) specification, (ii) estimation, (iii) forecasting and (iv) simulation are available for univariate and multivariate time series. The general nomenclature for the functions when we consider univariate time series is "`UniGAS...()`" and that for multivariate time series is "`MultiGAS...()`" .

In the R package **GAS**, several datasets are also included for reproducibility purposes, such as: US inflation (`cpichg`), US unemployment rate (`usunp`), realized volatility of the S&P500 Index (`sp500rv`) and intraday bid and ask quotes for Citygroup corporation (`tqdata`). These datasets are freely available online; see the R documentation for references. In this section, we use the monthly US inflation measured as the logarithmic change in the CPI available from the Federal Reserve Bank of St. Louis website https://fred.stlouisfed.org/. This dataset can be easily loaded in the R workspace using: `data("cpichg", package = "GAS")`.

3.1. Specification

Specification of GAS models is the first step the user needs to undertake. This is achieved by using the `UniGASSpec()` and `MultiGASSpec()` functions, in the cases of univariate and multivariate models, respectively. Both functions accept three arguments and return an object of class `uGASSpec` and `mGASSpec`, respectively. The three arguments are:

- `Dist`: A `character` indicating the name of the conditional distribution assumed for the data. Available distributions can be displayed using the function `DistInfo()` and are reported in Table 1. By default `Dist = "norm"`, *i.e.*, the Gaussian distribution.

- `ScalingType`: A `character` indicating the scaling mechanism for the conditional score, *i.e.*, the value of the γ parameter in (3). Possible choices are `"Identity"` ($\gamma = 0$), `"Inv"` ($\gamma = 1$) and `"InvSqrt"` ($\gamma = \frac{1}{2}$). For some distributions, only `ScalingType =`

`"Identity"` is supported; see function `DistInfo()` and Table 1. By default `ScalingType = "Identity"`, *i.e.*, no scaling occurs.[6]

- `GASPar`: A named `list` with `logical` entries containing information about which parameters of the conditional distribution have to be time–varying. Generally, each univariate distribution is identified by a series of maximum five parameters. These are indicated by `location, scale, skewness, shape` and `shape2`. Note that, for some distributions, these labels are not strictly related to their literal statistical meaning. Indeed, for the Exponential distribution `exp`, the term `location` indicates the usual intensity rate parameter; see the `DistInfo()` function and the R documentation for more details. For multivariate distributions, the set of parameters is indicated by `location, scale, correlation` and `shape`. For example, in the case of a multivariate Student–t distribution with mean vector $\boldsymbol{\mu}_t$, scale matrix $\boldsymbol{\Sigma}_t \equiv \mathbf{D}_t\mathbf{R}_t\mathbf{D}_t$, where $\mathbf{D}_t$ is the diagonal matrix of scales and $\mathbf{R}_t$ is the correlation matrix, and ν_t degrees of freedom, we have that: `location` refers to $\boldsymbol{\mu}_t$, `scale` refers to $\mathbf{D}_t$, `correlation` refers to $\mathbf{R}_t$ and `shape` refers to ν_t. By default, `GASPar = list(location = FALSE, scale = TRUE, skewness = FALSE, shape = FALSE)` for the univariate case, and `GASPar = list(location = FALSE, scale = TRUE, correlation = FALSE, shape = FALSE, shape2 = FALSE)` for the multivariate case.

The function `MultiGASSpec()` also accepts the additional `logical` argument `ScalarParameters` controlling for the parametrization of $\mathbf{A}$ and $\mathbf{B}$ in (4). By setting the argument `ScalarParameters = TRUE` (the default value), the coefficients controlling the evolution of the location, scale and correlation parameters are constrained to be the same across each group. For example, if $\mathbf{y}_t \in \Re^3$ follows a GAS process with conditional multivariate Gaussian distribution, the vector of time–varying parameters is $\boldsymbol{\theta}_t = (\mu_{1,t}, \mu_{2,t}, \mu_{3,t}, \sigma_{1,t}, \sigma_{2,t}, \sigma_{3,t}, \rho_{21,t}, \rho_{31,t}, \rho_{32,t})'$. If `ScalarParameters = TRUE`, the matrix of coefficients $\mathbf{A}$ is parameterized as:

$$\mathbf{A} \equiv \text{diag}\left(a_\mu, a_\mu, a_\mu, a_\sigma, a_\sigma, a_\sigma, a_\rho, a_\rho, a_\rho\right) ,$$

while, if `ScalarParameters = FALSE`, the matrix of coefficients $\mathbf{A}$ takes the form:

$$\mathbf{A} \equiv \text{diag}\left(a_{\mu_1}, a_{\mu_2}, a_{\mu_3}, a_{\sigma_1}, a_{\sigma_2}, a_{\sigma_3}, a_{\rho_{21}}, a_{\rho_{31}}, a_{\rho_{32}}\right) .$$

Hence, in the latter case, each element of $\boldsymbol{\theta}_t$ evolves heterogeneously with respect to the others. The same constraints are applied to $\mathbf{B}$, which means that, if `ScalarParameters = TRUE`, for the general N case, the number of parameters decreases from $3N\left(N+1\right)/2$ to $N\left(N+1\right)/2+2$. Additional constraints are introduced through the `GASPar` argument as in the univariate case; see `help("MultiGASSpec")`.

As an illustration, assume that we want to specify a Student–t GAS model with time–varying conditional mean and scale parameters, but fixed degree of freedom, *i.e.*, $\nu_t = \nu$. This can be easily done with the following lines of code:

```
R> GASSpec <- UniGASSpec(Dist = "std", ScalingType = "Identity",
+    GASPar = list(location = TRUE, scale = TRUE, shape = FALSE))
```

[6] In the R package **GAS**, the information matrices and the scores are always computed using their analytical formulations.

Label	Name	Type	Parameters	#	Scaling Type
norm	Gaussian	univariate	location, scale	2	Identity, Inv, InvSqrt
snorm (i)	Skew-Gaussian	univariate	location, scale, skewness	3	Identity
std (ii)	Student-t	univariate	location, scale, shape	3	Identity, Inv, InvSqrt
sstd (i)	Skew-Student-t	univariate	location, scale, skewness, shape	4	Identity
ast (iii)	Asymmetric Student-t with two tail decay parameters	univariate	location, scale, skewness, shape, shape2	5	Identity, Inv, InvSqrt
ast1 (iv)	Asymmetric Student-t with one tail decay parameter	univariate	location, scale, skewness, shape	4	Identity, Inv, InvSqrt
ald (v)	Asymmetric Laplace Distribution	univariate	location, scale, skewness	3	Identity, Inv, InvSqrt
ghskt (i)	Generalized Hyperbolic Skew Student-t Distribution	univariate	location, scale, skewness, shape	4	Identity
poi (vi)	Poisson	univariate	location	1	Identity, Inv, InvSqrt
gamma	Gamma	univariate	scale, shape	2	Identity, Inv, InvSqrt
exp (vii)	Exponential	univariate	location	1	Identity, Inv, InvSqrt
beta (viii)	Beta	univariate	scale, shape	2	Identity, Inv, InvSqrt
negbin	Negative Binomial	univariate	location, scale	2	Identity, Inv, InvSqrt
skellam (ix)	Skellam	univariate	location, scale	2	Identity
mvnorm	Multivariate Gaussian	multivariate	location, scale, correlation	9 (x)	Identity
mvt	Multivariate Student-t	multivariate	location, scale, correlation, shape	10 (x)	Identity

Table 1: Statistical distributions for which the R package **GAS** provides the functionality to simulate, estimate and forecast the time–variation in its parameters. The fifth column, #, reports the number of parameters of the distribution.
Notes: (i) the reparameterized Skew–Gaussian, Skew–Student–t and Generalized Hyperbolic Skew–Student–t for which the location and scale parameters coincide with the mean and the standard deviation of the distribution as done in the R package **rugarch** (Ghalanos 2015b), (ii) the usual Student–t distribution (not reparameterized in terms of the variance parameter), (iii) the Asymmetric Student–t distribution of Zhu and Galbraith (2010), (iv) the Asymmetric Student–t distribution of Zhu and Galbraith (2010) with equal tail decay parameters, (v) the `ald` distribution with the θ, σ, κ reparameterization, as specified in Kotz *et al.* (2001), (vi) for the Poisson distribution `location` means the usual intensity parameter, (vii) for the Exponential distribution `location` means the usual rate parameter, (viii) for the Beta distribution `shape` means the usual α parameter and `scale` means the usual β parameter, (ix) The Skellam distribution reparameterized in terms of mean and variance, (x) for $N = 3$.

Details about the object returned from `UniGASSpec()` are printed to the console by simply calling `GASSpec`:

```
R> GASSpec

------------------------------------------------------
-            Univariate GAS Specification            -
------------------------------------------------------
Conditional distribution
------------------------------------------------------
Name: Student-t
Label: std
Type: univariate
Parameters: location, scale, shape
Number of Parameters: 3
References:
------------------------------------------------------
GAS specification
------------------------------------------------------
Score scaling type:  Identity
Time varying parameters:  location, scale
------------------------------------------------------
```

Since the scaling matrix $\mathbf{S}_t$ is set to the identity matrix (*i.e.*, `ScalingType = "Identity"`) this model for the conditional Student–t distribution corresponds to the one described in Appendix A. Multivariate GAS specifications are analogously specified using the `MultiGASSpec()` function; see `help("MultiGASSpec")`.

3.2. Estimation

Similar to model specification, estimation is handled with two different functions for univariate and multivariate models: `UniGASFit()` and `MultiGASFit()`, respectively. These functions require two arguments: the GAS specification object `GASSpec` and the data, and returns an object of class `uGASFit` or `mGASFit`. By default, the optimization relies on the standard `optim` optimizer with `method = "BFGS"`. An additional optional `function` argument, called `fn.optimize`, can be provided by the user in order to rely on a different optimization procedure. This `function` must satisfy specific requirements; see the documentation manual for an example using Differential Evolution implemented in the **DEoptim** package (Mullen *et al.* 2011). Users can also employ the General Nonlinear Augmented Lagrange Multiplier method of Ye (1988) available in the R package **Rsolnp** (Ghalanos and Theussl 2015) by specifying `fn.optimize = fn.solnp`, where `fn.solnp` is a wrapper to the `solnp` function in **Rsolnp**.[7]

In the R package **GAS**, starting values for the optimizer are chosen in the following way: (i) estimate the long–term mean of $\boldsymbol{\theta}_t$ under the static version of the model (*i.e.*, with $\mathbf{A} = \mathbf{0}$ and $\mathbf{B} = \mathbf{0}$), and (ii) perform a grid search for the coefficients contained in $\mathbf{A}$ and $\mathbf{B}$ and whereby $\boldsymbol{\kappa}$ is set in such a way that the dynamic version of the model is still mean–reverting around the long–term mean of $\boldsymbol{\theta}_t$ estimated in the first step.

As an illustration, let us estimate the GAS model previously specified using the US inflation data included in the R package **GAS**:

```
R> data("cpichg", package = "GAS")
R> Fit <- UniGASFit(GASSpec, cpichg)
```

The computational time is less than one second on a modern computer (see the Section Computational Details for more details). Results can be inspected by calling the object `Fit`.[8]

```
R> Fit

----------------------------------------
-          Univariate GAS Fit          -
----------------------------------------

Model Specification:
T =  276
Conditional distribution:  std
Score scaling type:  Identity
Time varying parameters:  location, scale
```

[7] A previous version of this paper used **Rsolnp** as the default optimizer. As noted by a referee, the **Rsolnp** optimizer may lead to small differences across operating platforms.

[8] The command `summary(Fit)` provides the same information in addition to the analysis of the residuals as in the **fGarch** package (Wuertz *et al.* 2016).

```
-----------------------------------------
Estimates:
       Estimate Std. Error t value  Pr(>|t|)
kappa1  0.03736    0.03110   1.201 1.148e-01
kappa2 -0.25993    0.14084  -1.846 3.248e-02
kappa3 -2.84547    0.79650  -3.572 1.768e-04
a1      0.07173    0.01846   3.887 5.077e-05
a2      0.45372    0.21392   2.121 1.696e-02
b1      0.94317    0.02723  34.634 0.000e+00
b2      0.85560    0.07430  11.515 0.000e+00

-----------------------------------------
Unconditional Parameters:
location    scale    shape
  0.6575   0.1653   6.5261

-----------------------------------------
Information Criteria:
   AIC    BIC     np    llk
 370.4  395.8    7.0 -178.2

-----------------------------------------
Convergence:        0
-----------------------------------------

Elapsed time: 0.02 mins
```

The output printed to the console is divided into: (i) the summary of the model, (ii) the estimated coefficients along with significance levels according to their asymptotic Gaussian distribution, (iii) the long–term value of the time–varying parameters, *i.e.*, $\Lambda((\mathbf{I}-\widehat{\mathbf{B}})^{-1}\widehat{\boldsymbol{\kappa}})$, (iv) AIC and BIC information criteria in addition to the number of estimated parameters (`np`) and the log–likelihood (`llk`) evaluated at its optimum, (v) the convergence flag, and (vi) the computation time.[9]

Concerning the estimated coefficients, `kappa1`, `kappa2` and `kappa3` are the elements of vector $\boldsymbol{\kappa}$ in (9), *i.e.*, κ_μ, κ_ϕ and κ_ν, respectively. Analogously, `a1` and `a2` are the estimates of a_μ and a_ϕ and `b1` and `b2` are estimates of b_μ and b_ϕ, where ϕ refers to the scale parameter of the Student–t distribution; see Appendix A. Note that, since we have specified `scale = FALSE` in the `UniGASSpec()` function, coefficients `a3` and `b3`, corresponding to a_ν and b_ν are not reported (and constrained to zero during the optimization).

The R package **GAS** provides several methods to extract the relevant estimated quantities for objects of class `uGASFit` or `mGASFit`. They allow us to: (i) calculate several quantities of the estimated conditional distribution at each point in time, such as: quantiles, conditional moments and filtered parameters (see `quantile(Fit)`, `getMoments(Fit)` and

[9] Convergence flag follows the nomenclature of the default `optim` optimizer, that is: solver has converged (0), or not (1 or 2); see `help("optim")`. For user–defined optimizers, the definition of the convergence flag is flexible. The only requirement is that successful convergence is indicated with 0.

`getFilteredParameters(Fit)`, respectively), (ii) extract the estimated coefficients (`coef(Fit)`), (iii) generate a graphical representation of the results (`plot(Fit)`); see `help("uGASFit")` for details.

3.3. Forecasting

Forecasting is a crucial aspect in applied time series analysis and discussed in detail in the framework of GAS models by Blasques *et al.* (2016a). Given the parametric assumption of GAS models, predictions are usually given in the form of density forecasts, *i.e.*, the distribution of $\mathbf{y}_{T+h}|\mathbf{y}_{1:T}$ for $h \geq 1$. Knowing the predictive density, practitioners can extract any relevant quantities such as future expected value $\mathsf{E}_T[\mathbf{y}_{T+h}]$ or (co–)variance $\mathsf{V}_T[\mathbf{y}_{T+h}]$. For GAS models, the one–step ahead predictive distribution ($h = 1$) is analytically available while it needs to be estimated by simulation in the multi–step ahead case ($h > 1$).[10]

The R package **GAS** can handle both one–step and multi–step ahead forecasts. Consistent with previous nomenclature, functions for univariate and multivariate predictions are `UniGASFor()` and `MultiGASFor()`, respectively. These functions accept an object of class `uGASFit` or `mGASFit`, created using the functions `UniGASFit()` and `MultiGASFit()`, and return an object of class `uGASFor` and `mGASFor`, respectively. Additional arguments are:

- `H`: a `numeric` integer value representing the forecast horizon, *i.e.*, h. By default `H = 1`.
- `B`: a `numeric` integer value representing the number of draws to approximate the multi–step ahead predictive distribution when $h > 1$. By default `B = 1e4`.
- `ReturnDraws`: a `logical` argument controlling if the simulated draws from $\mathbf{y}_{T+1}|\mathbf{y}_{1:T}$, $\mathbf{y}_{T+2}|\mathbf{y}_{1:T}, \ldots, \mathbf{y}_{T+h}|\mathbf{y}_{1:T}$ have to be returned. By default `ReturnDraws = FALSE`.

Other arguments to perform rolling–type of forecasts are detailed in the documentation; see `help("UniGASFor")`. Practically, if we want to predict the next–year inflation (*i.e.*, $h = 12$ with the monthly series `cpichg`), after having estimated the GAS model of Section 3.2, we can execute the following code:

```
R> set.seed(123)
R> Forecast <- UniGASFor(Fit, H = 12)
```

and inspect the results by calling the object `Forecast`:

```
R> Forecast

-----------------------------------------
-        Univariate GAS Forecast        -
-----------------------------------------

Model Specification
Conditional distribution:  std
Score scaling type:  Identity
```

[10] As multi–step ahead forecasts are based on simulations, the user needs to set the seed of the random number generator for reproducibility.

```
Horizon:  12
Rolling forecast:  FALSE
----------------------------------------
Parameters forecast:
    location  scale shape
T+1  0.10130 0.1524 6.526
T+2  0.09499 0.1737 6.526
T+3  0.09382 0.2151 6.526
T+4  0.09256 0.2577 6.526
T+5  0.08747 0.3020 6.526

.................
     location  scale shape
T+8   0.08345 0.4219 6.526
T+9   0.07792 0.4575 6.526
T+10  0.07383 0.4900 6.526
T+11  0.07558 0.5199 6.526
T+12  0.07507 0.5465 6.526
```

which returns some model information and the predictions of future model parameters based on averages over `B` draws. `Forecast` is an object of class `uGASFor` which comes with several methods to extract and visualize the results; see `help("uGASFor")`.

As commonly done in time series analysis, predictions are generated from models fitted to rolling windows. The R package **GAS** includes this functionality via `UniGASRoll()` and `MultiGASRoll()`. These functions accept several arguments that we describe below in the univariate case:

- `data`: a vector of length T+`ForecastLength` containing all the observations.
- `GASSpec`: an object of class `uGASSpec` created with `UniGASSpec()`.
- `ForecastLength`: a `numeric` integer which specifies the length of the out–of–sample.
- `RefitEvery`: a `numeric` integer of periods before coefficients are re–estimated.
- `RefitWindow`: a `character` for the type of the window. As in the R package **rugarch** (Ghalanos 2015b), we define the options: `RefitWindow = "recursive"` and `RefitWindow = "moving"`. If `RefitWindow = "recursive"` all past observations are used when the model is re–estimated. If `RefitWindow = "moving"`, initial observations are eliminated.[11] We refer the reader to Marcellino *et al.* (2006) for a discussion about the difference between the "recursive" and the "moving" window approaches.

Other arguments useful to tailor the forecasting procedure and to parallelize the code execution are available and detailed in the R documentation; see `help("UniGASRoll")`.

Suppose now we are interested in assessing the forecasting performance of the GAS model with a Student–t conditional distribution and time–varying location and scale parameters,

[11] Note that the "moving window" approach is also referred to as the "rolling window" approach in the literature.

detailed in Appendix A, and specified in the object `GASSpec` in Section 3.1. We treat the last 150 observations of `cpichg` as out–of–sample and run a rolling–window forecast exercise using the following portion of code:

```
R> Roll <- UniGASRoll(cpichg, GASSpec, ForecastLength = 150,
+    RefitEvery = 3, RefitWindow = "moving")
```

where model coefficients are re–estimated quarterly (*i.e.*, every three observations with monthly data) using a moving windows (`RefitWindow = "moving"`). The code automatically makes a series of one–step ahead rolling predictions according to the model estimated using only the past information. This way, the user can perform out–of–sample analysis with GAS models. The object `Roll` belongs to the class `uGASRoll` which, as `uGASFit` and `uGASFor`, comes with several methods to extract and represent the results; see `help("uGASRoll")`.

3.4. Simulation

Simulation of univariate and multivariate GAS models is straightforward with the R package **GAS**. This can be easily done via `UniGASSim()` and `MultiGASSim()`; see the R documentation. Several examples, are reported in the `tests/testthat/test_Simulate.R` file included in the package tarball.

There are two possibilities for simulating GAS models. The first is to simulate from an estimated `uGASFit` or `mGASFit` object. For instance, if `Fit` is an `uGASFit` object delivered by the `UniGASFit()` function, the code `Sim <- UniGASSim(Fit, T.Sim = 1000)` simulates 1,000 observations from the corresponding GAS model. The second possibility is to fully specify a GAS model which means: (i) selecting the conditional distribution of the time series process, and (ii) specify the static parameters $\boldsymbol{\xi}$ governing the dynamics in $\boldsymbol{\theta}_t$. Regarding the former point, the vector $\boldsymbol{\kappa}$ and the system matrices $\mathbf{A}$ and $\mathbf{B}$ need to be specified. It is worth stressing that the definition of $\boldsymbol{\kappa}$ can be tricky, especially for multivariate models. In the analysis of time–varying parameter models, it is common to define $\boldsymbol{\kappa}$, $\mathbf{A}$ and $\mathbf{B}$ in such a way that the time–varying parameter $\boldsymbol{\theta}_t$ is covariance–stationary and that its unconditional expectation equals a target value. This is straightforward to do when the mapping function is linear, see, *e.g.*, Francq *et al.* (2011). When the link function is non–linear, which is the most common case in GAS modeling, it is more complex. The difficulty emerges from the fact that $\boldsymbol{\kappa}$ determines the unconditional value of $\widetilde{\boldsymbol{\theta}}_t$, that is $\mathsf{E}[\widetilde{\boldsymbol{\theta}}_t] = (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\kappa}$, implying that if the user wants to specify the model in terms of a target value $\boldsymbol{\theta}^* = \Lambda((\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\kappa})$, she needs to know the inverse of the mapping function $\Lambda(\cdot)$.[12] To address this problem, the functions `UniUnmapParameters()` and `MultiUnmapParameters()`, representing $\Lambda^{-1}(\cdot)$, are available for univariate and multivariate models, respectively. This way, the user can easily specify $\boldsymbol{\kappa}$ such that $\boldsymbol{\kappa} \equiv (\mathbf{I} - \mathbf{B})\Lambda^{-1}(\boldsymbol{\theta}^*)$. Table 2 lists the numerical bounds imposed for the univariate distributions, such that the arguments of `UniUnmapParameters()` cannot take values outside those ranges. For the multivariate case, please refer to the examples reported in the `inst/test/SimulateGAS.R` file included in the package tarball.

[12] Here we define the "target value" as the unconditional expectation of the time–varying parameter the user has in mind. This targeting approach requires the time–varying parameter model to be stationary, as explained in, *e.g.*, Blasques *et al.* (2014c).

Label	location	scale	skewness	shape
norm	$\Re$	$\Re^+$	–	–
snorm	$\Re$	$\Re^+$	$(0.5, 1.5)$	–
std	$\Re$	$\Re^+$	–	$(4.01, 50.0)$
sstd	$\Re$	$\Re^+$	$(0.5, 1.5)$	$(4.01, 50.0)$
ast1 (i)	$\Re$	$\Re^+$	$(0.01, 0.99)$	$(4.01, 50.0)$
ald	$\Re$	$\Re^+$	$\Re^+$	–
ghskt	$\Re$	$\Re^+$	$\Re$	$(4.01, 50.0)$
poi	$\Re^+$	–	–	–
gamma	$\Re^+$	$\Re^+$	–	–
exp	$\Re^+$	–	–	–
beta	$\Re^+$	$\Re^+$	–	–
negbin	$(0, 1)$	$\Re^+$	–	–
skellam	$\Re$	$\Re^+$	–	–

Table 2: Overview of the restrictions on the allowed values for the parameters of the univariate distributions, for which the R package **GAS** provides the functionality to simulate, estimate and forecast the time variation in the parameters. When the parameter space is $\Re^+$, we use the exponential link function reported in (10) with $c = 0$, while when the space is of the type (a, b), we use the modified logistic link function reported in (11); see Appendix B.
Notes: (i) for ast the same constraints apply, and shape2 is constrained in $(2.01, 50.0)$.

Suppose we want to simulate $T = 1,000$ observations from the Student–t GAS model reported in Appendix A with time–varying location and scale, but constant shape parameters. Assume our target value for the parameters is $\boldsymbol{\theta}^* = (\mu^*, \sigma^*, \nu^*)'$ with $\mu^* = 0.1, \sigma^* = 1.5$ and $\nu^* = 7$. The matrix $\mathbf{A}$ and $\mathbf{B}$ are defined as:

$$\mathbf{A} = \text{diag}\,(0.1, 0.4, 0.0)$$
$$\mathbf{B} = \text{diag}\,(0.9, 0.95, 0.0)\,,$$

such that both the conditional mean and the conditional variance evolve quite persistently over time, while the shape parameter is constant. The implementation of UniUnmapParameters() and UniGASSim() proceeds as:

```
R> A <- diag(c(0.1, 0.4, 0.0))
R> B <- diag(c(0.9, 0.95, 0.0))
R> ThetaStar <- c(0.1, 1.5, 7.0)
R> kappa <- (diag(3) - B) %*% UniUnmapParameters(ThetaStar, "std")
R> Sim <- UniGASSim(T.sim = 1000, kappa = kappa, A = A,
+    B = B, Dist = "std", ScalingType = "Identity")
```

where Sim is an object of class uGASSim which comes with several methods such as show, plot, and getMoments, among others; see help("uGASSim").

4. Application to financial returns

In order to illustrate how the R package **GAS** can be used in practical situations, we present an empirical application with univariate and multivariate time series of financial returns. We consider daily log–returns (in percentage points) of the Dow Jones 30 constituents available in the `dji30ret` dataset. This dataset includes the closing value log–returns from March 3rd, 1987 to February 3rd, 2009 for a total of 5,521 observations per series. The dataset can be easily loaded in the workspace using:

```
R> library("GAS")
R> data("dji30ret", package = "GAS")
```

where `dji30ret` is a 5521×30 `data.frame` containing the daily log–returns in percentage points. Our analysis is a typical out–of–sample exercise, meaning that: (i) we estimate the models using an in–sample period, (ii) we do rolling one–step ahead predictions of the conditional distribution for the observations in the out–of–sample period, and (iii) that we compare the models according to their out–of–sample performance.

The models we consider are univariate/multivariate GAS models estimated with the R package **GAS**, and univariate/multivariate GARCH models estimated using the popular R packages **rugarch** (Ghalanos 2015b) and **rmgarch** (Ghalanos 2015a), respectively. The univariate specifications we consider are: (i) the Skew–Student–t GAS model with only time–varying scale parameter (*i.e.*, `Dist = "sstd"`) and, (ii) the GARCH(1,1) model with Skew–Student–t distributed error. For both models we employ the Skew–Student–t distribution of Fernández and Steel (1998) reparametrised such that the location and scale parameters coincide with the mean and the standard deviation of the distribution as done in the **rugarch** package.

For the multivariate specifications, we consider: (i) the GAS model with conditional multivariate Student–t distribution with time–varying scales and correlations used in Creal *et al.* (2011) and, (ii) the Dynamic Conditional Correlation (DCC) model of Engle (2002) with a conditional multivariate Student–t distribution. For simplicity, the multivariate analysis only considers three return series, namely the returns for: Caterpillar Inc. (CAT), 3M (MMM) and Pfizer Inc. (PFE).

The code used to specify the univariate and multivariate GAS models is:

```
R> uGASSpec <- UniGASSpec(Dist = "sstd", ScalingType = "Identity",
+    GASPar = list(scale = TRUE))
```

and:

```
R> mGASSpec <- MultiGASSpec(Dist = "mvt", ScalingType = "Identity",
+    GASPar = list(scale = TRUE, correlation = TRUE))
```

respectively.

The last $H = 3,000$ observations (from January 27th, 1991, to the end of the sample) compose the out–of–sample period. During the out–of–sample period, one–step ahead density predictions are constructed by the univariate and multivariate specifications. The models (and therefore coefficients) are re–estimated using a moving–window every hundredth observations, as detailed in Section 3.3. One–step ahead rolling prediction are then computed as:

```
R> library("parallel")
R> cluster <- makeCluster(8)
R>
R> luGASRoll <- list()
R> N = ncol(dji30ret)
R> for (i in 1:N) {
+   luGASRoll[[i]] <- UniGASRoll(data = dji30ret[, i],
+     GASSpec = uGASSpec, ForecastLength = 3000,
+     RefitEvery = 100, cluster = cluster)
+ }
R> names(luGASRoll) <- colnames(dji30ret)
```

and:

```
R> mGASRoll <- MultiGASRoll(data = dji30ret[, c("CAT", "MMM", "PFE")],
+    GASSpec = mGASSpec, ForecastLength = 3000,
+    RefitEvery = 100, cluster = cluster)
R> stopCluster(cluster)
```

for the univariate and multivariate cases, respectively. To speed up the computations, we run the code over eight processors via the R package **parallel**. Again, we emphasize that in the case of one–step ahead density predictions, results are available in closed–form, in contrast with multi–step ahead forecasts which are based on simulations. Hence, our results do not depend on the seed or the parallelization scheme used for the computations.

Let us now compare the ability of GAS and GARCH models in predicting the one–step ahead distribution using so–called *scoring rules*, which compare the predicted density with the ex–post realized value of the return and deliver a score which defines a ranking across the alternative models at each point in time (Gneiting *et al.* 2007). Generally, we define $S_{t+1} \equiv S(y_{t+1}, p(y_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}))$ as the score at time $t+1$ for having predicted $p(y_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1})$ when y_{t+1} has been realized. We consider two widely used scoring rules:

- The average weighted Continuous Ranked Probability Score (wCRPS):

$$\overline{wCRPS} \equiv \frac{1}{H} \sum_{t=T}^{T+H-1} \int_{-\infty}^{\infty} w(z) \left(F\left(z; \widehat{\boldsymbol{\theta}}_{t+1}\right) - \mathbb{I}_{\{y_{t+1}<z\}} \right)^2 \mathrm{d}z, \tag{6}$$

  where $w(z)$ is a weight function that emphasizes regions of interest of the predictive distribution, such as the tails or the center and $F\left(z; \widehat{\boldsymbol{\theta}}_{t+1}\right)$ is the predictive cumulative density function evaluated in z.[13] Similarly to Gneiting and Ranjan (2011), we consider the cases of: (i) a weighting that gives equal emphasis to all the parts of the distribution; $w(z) = 1$, (ii) a weighting that focuses on the center; $w(z) = \phi_{a,b}(z)$; (iii) a weighting that focuses on the tails; $w(z) = 1 - \phi_{a,b}(z) / \phi_{a,b}(0)$, (iv) a weighting that focuses on the right tail; $w(z) = \Phi_{a,b}(z)$, and (v) a weighting that focuses on the left tail $w(z) = 1 - \Phi_{a,b}(z)$. The functions $\phi_{a,b}(z)$ and $\Phi_{a,b}(z)$ are the *pdf* and *cdf* of a Gaussian distribution with mean a and standard deviation b, respectively. The label `uniform` represents the case where equal emphasis is given to all the parts of the distribution.

[13] We denote by $\mathbb{I}_{\{\cdot\}}$ the indicator function which equals one if the condition is satisfied and zero otherwise.

- The average Negative Log Score (NLS):

$$\overline{NLS} \equiv -\frac{1}{H}\sum_{t=T}^{T+H-1} \log p(y_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}). \tag{7}$$

  Consistent with Gneiting *et al.* (2007), we specify the Negative Log Score such that the "direction" between the two scoring rules is the same, *i.e.*, forecasts with lower $\overline{NLS}$ and lower $\overline{wCRPS}$ are preferred.

The two aforementioned scoring rules can be easily evaluated using the `BacktestDensity()` function available in the R package **GAS**. The `BacktestDensity()` function accepts an object of class `uGASRoll`, and returns a list with two elements: (i) the average wCRPS and average NLS as in (6) and (7), and (ii) their values at each point in time. To evaluate the integral in (6), we use the numerical integration scheme adopted by Gneiting and Ranjan (2011). To this end, the `BacktestDensity()` function accepts the following additional arguments:

- `lower`: `numeric` representing the lower bound of the numerical integration.
- `upper`: `numeric` as `lower` but for the upper bound.[14]
- `K`: `numeric` integer representing the number of points used in the numerical integration.[15] By default `K = 1000`,

plus the two `numeric` arguments, `a` and `b`, representing a and b in the weight functions. By default, `a` and `b` are equal to the empirical mean and standard deviation of the in–sample data, respectively.[16]

In our case, in order to evaluate $\overline{NLS}$ and $\overline{wCRPS}$ for the first asset, we can simply run:[17]

```
R> DensityBacktest <- BacktestDensity(luGASRoll[[1]],
+    lower = -100, upper = 100)
R> DensityBacktest$average

    NLS uniform  center   tails  tail_r  tail_l
 2.2163  1.3292  0.1767  0.5251  0.6522  0.6770
```

Table 3 reports the test statistics for the Diebold and Mariano (1995) (DM) test of equal performance between the series of Negative Log Scores and weighed Continuous Ranked Probability Scores for univariate GAS and GARCH models across the out–of–sample period.[18]

[14] The two arguments `lower` and `upper` coincide with y_l and y_u in Equation 16 of Gneiting and Ranjan (2011), respectively. These are two numeric objects with no default value, *i.e.*, the user has to define these values according to her research design.

[15] Equals to I in Equation 16 of Gneiting and Ranjan (2011).

[16] These values can be chosen in order to target some "optimal" prediction level, or to add more flexibility and focus on specific parts the predictive distribution; see Gneiting and Ranjan (2011).

[17] Chosen `lower` and `upper` values define a proper range for log–returns in percentage points as the one considered here.

[18] Results of Table 3 were obtained on a Windows platform with the setup described in the Section "Computational details". When computations were performed on Linux or Mac OS, the test statistics matched up to two digits, except for a couple of entries (up to one digit) and lead to the same conclusions in terms of relative performance of the GAS and GARCH models. We thank a referee for having noted these differences across platforms. For more details, we refer the reader to the Section "Computational details".

Asset	NLS	Uniform	Center	Tails	Tails–r	Tails–l
AA	-1.98^{b}	-2.36^{a}	-2.15^{b}	-2.02^{b}	-2.16^{b}	-1.04
AXP	-2.98^{a}	-2.35^{a}	-3.14^{a}	-1.21	-1.53^{c}	-1.60^{c}
BA	-1.44^{c}	-1.91^{b}	-2.79^{a}	-1.13	-1.21	-1.39^{c}
BAC	-2.95^{a}	-1.44^{c}	-3.80^{a}	0.22	1.26	-4.00^{a}
C	-0.66	-0.55	-1.82^{b}	0.34	-1.59^{c}	0.40
CAT	-5.26^{a}	-5.20^{a}	-5.61^{a}	-3.77^{a}	-2.17^{b}	-5.29^{a}
CVX	0.84	1.26	0.00	1.41^{c}	0.71	1.04
DD	-1.09	-0.57	-0.53	-0.51	0.23	-0.91
DIS	-2.10^{b}	-1.69^{b}	-2.40^{a}	-1.00	-1.69^{b}	-0.84
GE	-3.29^{a}	-3.33^{a}	-3.76^{a}	-2.61^{a}	-3.44^{a}	-2.21^{b}
GM	0.08	-0.42	-1.22	-0.20	-1.03	0.39
HD	-3.39^{a}	-2.53^{a}	-1.84^{b}	-2.72^{a}	-1.44^{c}	-1.91^{b}
HPQ	-4.25^{a}	-4.50^{a}	-3.58^{a}	-4.53^{a}	-3.08^{a}	-3.50^{a}
IBM	-3.54^{a}	-3.79^{a}	-3.41^{a}	-3.57^{a}	-1.99^{b}	-3.47^{a}
INTC	-3.17^{a}	-1.71^{b}	-1.70^{b}	-1.39^{c}	-2.93^{a}	0.09
JNJ	-3.74^{a}	-2.15^{b}	-3.57^{a}	-0.69	-0.14	-3.05^{a}
JPM	-2.05^{b}	-2.12^{b}	-2.91^{a}	-1.21	-1.22	-1.55^{c}
AIG	1.07	0.66	-1.10	0.71	-0.79	0.96
KO	-3.12^{a}	-3.07^{a}	-3.78^{a}	-1.59^{c}	-3.39^{a}	-1.69^{b}
MCD	-2.03^{b}	-1.86^{b}	-2.21^{b}	-1.39^{c}	-1.67^{b}	-1.10
MMM	-4.15^{a}	-4.54^{a}	-4.13^{a}	-4.12^{a}	-3.33^{a}	-3.37^{a}
MRK	-2.98^{a}	-3.63^{a}	-4.15^{a}	-3.08^{a}	-4.56^{a}	-1.91^{b}
MSFT	-3.42^{a}	-2.74^{a}	-3.14^{a}	-1.64^{c}	-2.64^{a}	-1.69^{b}
PFE	-3.65^{a}	-3.54^{a}	-3.72^{a}	-3.07^{a}	-4.03^{a}	-2.46^{a}
PG	-1.78^{b}	-1.85^{b}	-2.91^{a}	-1.25	-1.34^{c}	-1.31^{c}
T	-0.41	-0.24	-1.19	0.26	0.35	-0.65
UTX	-1.50^{c}	-1.61^{c}	-1.74^{b}	-1.38^{c}	-1.10	-0.89
VZ	-1.99^{b}	-1.94^{b}	-1.50^{c}	-1.95^{b}	-1.35^{c}	-1.52^{c}
WMT	2.11^{b}	0.92	0.84	0.81	-0.42	1.68^{b}
XOM	0.15	0.38	-1.29^{c}	0.72	0.69	-0.04

Table 3: Test statistics for the Diebold and Mariano (1995) test of equal performance between the series of negative Log Scores and weighted Continuous Ranked Probability Scores for univariate GAS and GARCH models across the out–of–sample logarithmic returns in percentage points of Dow Jones 30 constituents. Negative values indicate that GAS models provide more accurate predictions of the one–step ahead conditional distribution while positive values favour GARCH. The apexes a, b and c represent rejection of the null hypothesis of Equal Predictive Ability at the $1\%, 5\%$ and 10% confidence levels, respectively. The out–of–sample period spans from January 27th, 1991, to February 3rd, 2009 for a total of 3,000 observations.

Negative values indicate that GAS models generate more accurate predictions of the one–step ahead conditional distribution while positive values favour GARCH. We find that, for almost all the series, GAS outperforms GARCH at very high confidence levels according to both NLS and wCRPS. Interestingly, our results suggest that GAS delivers more accurate results

whatever part of the conditional distribution the $\overline{wCRPS}$ emphasizes.

For the multivariate analysis we only consider $\overline{NLS}$. In this case, the DM test statistic is -3.34, which strongly favours the GAS model against the DCC specification. To further investigate this result, we report in Figure 1 the Cumulative sum of the differences between the Log Scores (CLS) of GAS and DCC defined as:

$$CLS_{T:T+l}^{GAS|DCC} \equiv \sum_{t=T}^{t=T+l-1} \log p\left(\mathbf{y}_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}^{GAS}\right) - \log p\left(\mathbf{y}_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}^{DCC}\right),$$

where $p\left(\mathbf{y}_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}^{GAS}\right)$ and $p\left(\mathbf{y}_{t+1}; \widehat{\boldsymbol{\theta}}_{t+1}^{DCC}\right)$ are the densities predicted from GAS and DCC evaluated in $\mathbf{y}_{t+1}$, respectively. The series of Log Scores for the multivariate GAS models is available in the output of the `BacktestDensity()` function, or can be extracted using the `LogScore` method defined for `mGASRoll` objects:

```
R> LS_MGAS <- LogScore(mGASRoll)
```

In Figure 1, periods when the time series line slopes upward represent periods in which GAS outperforms DCC, while downward–sloping segments indicate periods when the DCC forecast is more accurate. From this plot, we clearly understand the result of DM test. Interestingly, we find that GAS starts dominating DCC after 2003.

5. Conclusion

This article introduced the R package **GAS** for simulating, estimating and forecasting time–varying parameter models under the Generalized Autoregressive Score framework. It allows practitioners in many scientific areas to perform their applied research using GAS models in a user–friendly environment.

We introduced the model specification in a general way and illustrated the package usage. In particular, we performed an empirical application using financial data in which we compared the performance of univariate and multivariate GAS and GARCH models. Given the flexibility of GAS models and the availability of several statistical distributions in the **GAS** package, a number of different applications can be easily handled, such as: (i) the analysis of integer valued time series using the Poisson GAS model (`poi`), (ii) the analysis of (0,1)–bounded time series using the Beta GAS model (`beta`), (iii) the analysis of strictly positive time series with an inverse location/scale dependence using the Gamma GAS model (`gamma`).

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

Computational details

The results in this paper were obtained using R 3.4.1 (R Core Team 2017) with the packages: **GAS** version 0.2.3 (Catania *et al.* 2017), **lmtest** version 0.9-35 (Zeileis and Hothorn 2002), **MASS** version 7.3-47 (Venables and Ripley 2002; Ripley 2016), **numDeriv** version 2016.8-1 (Gilbert and Varadhan 2016), **Rcpp** version 0.12.12 (Eddelbuettel and François 2011; Eddelbuettel *et al.* 2017a), **RcppArmadillo** version 0.7.900.2.0 (Eddelbuettel and Sanderson 2014;

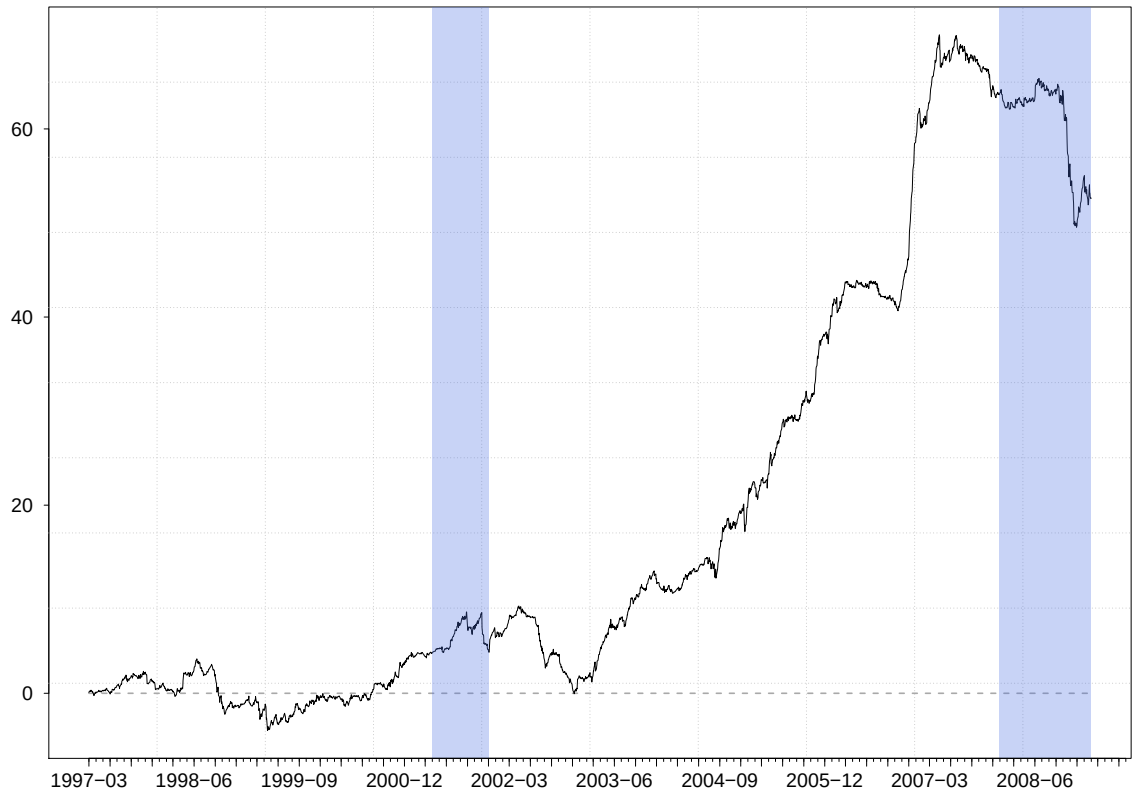

Figure 1: Cumulative out–of–sample Log Score differences between the multivariate Student–t GAS and the DCC(1,1) model of Engle (2002) with multivariate Student–t errors. Periods when the time series line slopes upward represent periods in which GAS outperforms GARCH, while downward–sloping segments indicate periods when the GARCH forecast is more accurate. The blue shaded area represents periods of recession in the US economy according to the "USREC" series available from the Federal Reserve Bank of St. Louis web site at https://fred.stlouisfed.org/series/USREC.

Eddelbuettel *et al.* 2017b), **rugarch** version 1.3-6 (Ghalanos 2015b), **sandwich** version 2.3-4 (Zeileis 2004), **xts** version 0.10-0 (Ryan and Ulrich 2014), and **zoo** version 1.8-0 (Zeileis and Grothendieck 2005). Some datasets available in the package were downloaded using the **quantmod** package (Ryan 2016), version 0.4-10. Computations were performed on Windows 10 Pro, x86_64-w64-mingw32/x64 (64-bit) with Intel(R) Xeon(R) CPUE3-1535M v6 Max Turbo Frequency 4.20 GHz. Full output of the computations is available in the file `sink_PART_II_R3.4.1_mingw32.txt` in the folder `inst/doc`.

R itself and all packages used are available from CRAN at http://CRAN.R-project.org/. The package **GAS** is available from the CRAN repository at https://cran.r-project.org/package=GAS. The version under development is available in GitHub at https://github.com/LeopoldoCatania/GAS.

The folder `inst/doc` inside the **GAS** package tarball contains additional technical documentations. A step by step guide on how to add a new statistical distribution in the **GAS** package is reported in the file `AddNewDistribution.pdf`. To ensure replication of the results in Table 3 across the various platforms we report, in the folder `inst/doc`, the estimation results of

Section 4 obtained under Windows 10 Pro, Linux Ubuntu 12.04.5 LTS and Mac OS X 10.12 Sierra. We further provide the average negative log–scores of the individual models (see files `mGAS.txt` and `mGARCH.txt`). Estimation results match up to seven digits for GAS models and up to five digits for GARCH model across the platforms.

Acknowledgments

The authors thank the Associate Editor (Paul Gilbert) and two referees for their comments. We also thank Alexios Ghalanos for his help regarding the platform dependence of the **Rsolnp** and **rugarch** packages. The authors acknowledge Google for financial support via the Google Summer of Code 2016 project "GAS". Any remaining errors or shortcomings are the authors' responsibility.

References

Andres P (2014). "Maximum Likelihood Estimates for Positive Valued Dynamic Score Models: The **DySco** Package." *Computational Statistics & Data Analysis*, **76**, 34–42. doi:10.1016/j.csda.2013.11.004.

Blasques F, Koopman SJ, Łasak K, Lucas A (2016a). "In–Sample Confidence Bands and Out–of–Sample Forecast Bands for Time–Varying Parameters in Observation–Driven Models." *International Journal of Forecasting*, **32**(3), 875–887.

Blasques F, Koopman SJ, Lucas A (2014a). "Maximum Likelihood Estimation for Correctly Specified Generalized Autoregressive Score Models: Feedback Effects, Contraction Conditions and Asymptotic Properties." *techreport TI 14-074/III*, Tinbergen Institute. URL http://www.tinbergen.nl/discussionpaper/?paper=2332.

Blasques F, Koopman SJ, Lucas A (2014b). "Maximum Likelihood Estimation for Generalized Autoregressive Score Models." *techreport TI 2014-029/III*, Tinbergen Institute. URL http://www.tinbergen.nl/discussionpaper/?paper=2286.

Blasques F, Koopman SJ, Lucas A (2014c). "Stationarity and Ergodicity of Univariate Generalized Autoregressive Score Processes." *Electronic Journal of Statistics*, **8**(1), 1088–1112. doi:10.1214/14-EJS924.

Blasques F, Koopman SJ, Lucas A, Schaumburg J (2016b). "Spillover Dynamics for Systemic Risk Measurement using Spatial Financial Time Series Models." *Journal of Econometrics*, **195**(2), 211–223. doi:http://dx.doi.org/10.1016/j.jeconom.2016.09.001.

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

Box GE, Jenkins GM (1970). *Time Series Analysis: Forecasting and Control*. Holden–Day, San Francisco.

Catania L, Billé AG (2017). "Dynamic Spatial Autoregressive Models with Autoregressive and Heteroskedastic Disturbances." *Journal of Applied Econometrics*, pp. 1–19. ISSN 1099-1255. doi:10.1002/jae.2565.

Catania L, Boudt K, Ardia D (2017). ***GAS**: Generalised Autoregressive Score Models*. R package version 0.2.3, URL https://cran.r-project.org/package=GAS.

Creal D, Koopman SJ, Lucas A (2011). "A Dynamic Multivariate Heavy–Tailed Model for Time–Varying Volatilities and Correlations." *Journal of Business & Economic Statistics*, **29**(4), 552–563. doi:10.1198/jbes.2011.10070.

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

Creal D, Schwaab B, Koopman SJ, Lucas A (2014). "Observation–Driven Mixed–Measurement Dynamic Factor Models with an Application to Credit Risk." *The Review of Economics and Statistics*, **96**(5), 898–915. doi:10.1162/REST_a_00393.

Diebold FX, Mariano RS (1995). "Comparing Predictive Accuracy." *Journal of Business & Economic Statistics*, **13**(3), 253–263. doi:10.1080/07350015.1995.10524599.

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

Eddelbuettel D, François R, Allaire J, Ushey K, Kou Q, Bates D, Chambers J (2017a). ***Rcpp**: Seamless R and C++ Integration*. R package version 0.12.12, URL https://cran.r-project.org/package=Rcpp.

Eddelbuettel D, François R, Bates D (2017b). ***RcppArmadillo**: **Rcpp** Integration for the **Armadillo** Templated Linear Algebra Library*. R package version 0.7.900.2.0, 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 (2002). "Dynamic Conditional Correlation: A Simple Class of Multivariate Generalized Autoregressive Conditional Heteroskedasticity Models." *Journal of Business & Economic Statistics*, **20**(3), 339–350. doi:10.1198/073500102288618487.

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

Francq C, Horváth L, Zakoïan JM (2011). "Merits and Drawbacks of Variance Targeting in GARCH Models." *Journal of Financial Econometrics*, **9**(4), 619. doi:10.1093/jjfinec/nbr004.

Ghalanos A (2015a). ***rmgarch**: Multivariate GARCH Models*. R package version 1.3-0, URL https://cran.r-project.org/package=rmgarch.

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

Ghalanos A, Theussl S (2015). ***Rsolnp**: General Non–Linear Optimization using Augmented Lagrange Multiplier Method*. R package version 1.16, URL https://cran.r-project.org/package=Rsolnp.

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

Gneiting T, Balabdaoui F, Raftery AE (2007). "Probabilistic Forecasts, Calibration and Sharpness." *Journal of the Royal Statistical Society B*, **69**(2), 243–268. doi:10.1111/j.1467-9868.2007.00587.x.

Gneiting T, Ranjan R (2011). "Comparing Density Forecasts using Threshold –and Quantile–Weighted Scoring Rules." *Journal of Business & Economic Statistics*, **29**(3), 411–422. doi:10.1198/jbes.2010.08110.

Harvey AC (2013). *Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series*. Cambridge University Press.

Harvey AC, Luati A (2014). "Filtering with Heavy Tails." *Journal of the American Statistical Association*, **109**(507), 1112–1122. doi:10.1080/01621459.2014.887011.

Harvey AC, Sucarrat G (2014). "EGARCH Models with Fat Tails, Skewness and Leverage." *Computational Statistics & Data Analysis*, **76**, 320–338. doi:10.1016/j.csda.2013.09.022.

Harvey AC, Thiele S (2016). "Testing Against Changing Correlation." *Journal of Empirical Finance*, **38, Part B**, 575–589. doi:10.1016/j.jempfin.2015.09.003.

Jaeckel P, Rebonato R (1999). "The Most General Methodology for Creating a Valid Correlation Matrix for Risk Management and Option Pricing Purposes." *Journal of Risk*, **2**(2), 17–28.

Janus P, Koopman SJ, Lucas A (2014). "Long Memory Dynamics for Multivariate Dependence under Heavy Tails." *Journal of Empirical Finance*, **29**, 187–206. doi:10.1016/j.jempfin.2014.09.007.

Kalman RE (1960). "A New Approach to Linear Filtering and Prediction Problems." *Transactions of the ASME–Journal of Basic Engineering D*, **82**, 35–45. doi:10.1115/1.3662552.

Kotz S, Kozubowski T, Podgorski K (2001). *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance*. Springer–Verlag, New York. URL http://www.springer.com/us/book/9780817641665.

Lucas A, Zhang X (2016). "Score–Driven Exponentially Weighted Moving Averages and Value–at–Risk Forecasting." *International Journal of Forecasting*, **32**(2), 293–302. doi:10.1016/j.ijforecast.2015.09.003.

Marcellino M, Stock JH, Watson MW (2006). "A Comparison of Direct and Iterated Multistep AR Methods for Forecasting Macroeconomic Time Series." *Journal of Econometrics*, **135**(1–2), 499–526. doi:10.1016/j.jeconom.2005.07.020.

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.

Oh DH, Patton AJ (2016). "Time–Varying Systemic Risk: Evidence from a Dynamic Copula Model of CDS Spreads." *Journal of Business & Economic Statistics*, **0**, 1–47. doi:10.1080/07350015.2016.1177535.

Pinheiro JC, Bates DM (1996). "Unconstrained Parametrizations for Variance–Covariance Matrices." *Statistics and Computing*, **6**(3), 289–296. doi:10.1007/BF00140873.

Pourahmadi M, Wang X (2015). "Distribution of Random Correlation Matrices: Hyperspherical Parameterization of the Cholesky Factor." *Statistics & Probability Letters*, **106**, 5–12. doi:10.1016/j.spl.2015.06.015.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. R version 3.4.1, URL https://www.R-project.org/.

Rapisarda F, Brigo D, Mercurio F (2007). "Parameterizing Correlations: A Geometric Interpretation." *IMA Journal of Management Mathematics*, **18**(1), 55–73. doi:10.1016/j.spl.2015.06.015.

Ripley B (2016). ***MASS**: Support Functions and Datasets for Venables and Ripley's MASS*. R package version 7.3-47, URL https://CRAN.R-project.org/package=MASS.

Ryan JA (2016). ***quantmod**: Quantitative Financial Modelling Framework*. R package version 0.4-10, URL https://CRAN.R-project.org/package=quantmod.

Ryan JA, Ulrich JM (2014). ***xts**: Extensible Time Series*. R package version 0.10-0, URL https://CRAN.R-project.org/package=xts.

Sanderson C (2010). "**Armadillo**: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments." *Technical report*, NICTA. URL http://arma.sourceforge.net/.

Shephard N (2005). *Stochastic Volatility: Selected Readings*. Oxford University Press, Oxford.

Sucarrat G (2013). "**betategarch**: Simulation, Estimation and Forecasting of Beta-Skew-t-EGARCH Models." *The R Journal*, **5**(2), 137–147. URL https://journal.r-project.org/archive/2013-2/sucarrat.pdf.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer–Verlag, New York. URL http://www.stats.ox.ac.uk/pub/MASS4.

Wuertz D, with contribution from Michal Miklovic YC, Boudt C, Chausse P, others (2016). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*. R package version 3010.82.1, URL https://CRAN.R-project.org/package=fGarch.

Ye Y (1988). *Interior Algorithms for Linear, Quadratic, and Linearly Constrained Convex Programming*. Ph.D. thesis, Stanford University.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. URL http://www.jstatsoft.org/v11/i10/.

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.

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." *R News*, **2**(3), 7–10. URL https://CRAN.R-project.org/doc/Rnews/.

Zhu D, Galbraith JW (2010). "A Generalized Asymmetric Student-t Distribution with Application to Financial Econometrics." *Journal of Econometrics*, **157**(2), 297–305. doi:10.1016/j.jeconom.2010.01.013.

A. The GAS model with conditional Student–t distribution

Let us consider the case where the distribution of the univariate random variable $y_t \in \Re$, conditionally on $\mathbf{y}_{1:t-1}$, is Student–t with location μ_t, scale $\phi_t \in \Re^+$, and $\nu_t > 2$ degrees of freedom[19], *i.e.*, $\boldsymbol{\theta}_t = (\mu_t, \phi_t, \nu_t)'$ and:

$$p(y_t; \boldsymbol{\theta}_t) \equiv \frac{\Gamma\left(\frac{\nu_t+1}{2}\right)}{\Gamma\left(\frac{\nu_t}{2}\right)\phi_t\sqrt{\pi\nu_t}}\left(1+\frac{(y_t-\mu_t)^2}{\nu_t\phi_t^2}\right)^{-\frac{\nu_t+1}{2}}. \tag{8}$$

As will become clear, the score corresponding to the Student–t distribution has the advantage of dampening the effect of extreme observations on the future volatility, when the Student–t has sufficiently fat tails. It has been used by Creal *et al.* (2013) and Lucas and Zhang (2016) under the name tGAS, and by Harvey (2013) and Harvey and Luati (2014) under the name Beta–t–EGARCH.

Differentiating the logarithm of (8) with respect to $\boldsymbol{\theta}_t$ leads to the score vector $\boldsymbol{\nabla}_t(y_t, \boldsymbol{\theta}_t) = (\nabla_t^\mu, \nabla_t^\phi, \nabla_t^\nu)^\top$, with:

$$\begin{aligned}
\nabla_t^\mu &\equiv \frac{(\nu_t+1)(y_t-\mu_t)}{\nu_t\phi_t\left(1+\frac{(y_t-\mu_t)^2}{\nu_t\phi_t}\right)} \\
\nabla_t^\phi &\equiv \frac{(\nu_t+1)(y_t-\mu_t)^2}{2\nu_t\phi_t^2\left(1+\frac{(y_t-\mu_t)^2}{\nu_t\phi_t}\right)} - \frac{1}{\phi_t} \\
\nabla_t^\nu &\equiv \frac{1}{2}\psi\left(\frac{\nu_t+1}{2}\right) - \frac{1}{2}\psi\left(\frac{\nu_t}{2}\right) - \frac{1}{2\nu_t} \\
&\quad - \frac{1}{2}\log\left(1+\frac{(y_t-\mu_t)^2}{\nu_t\phi_t}\right) + \frac{(\nu_t+1)(y_t-\mu_t)^2}{2\nu_t^2\phi_t\left(1+\frac{(y_t-\mu_t)^2}{\nu_t\phi_t}\right)},
\end{aligned}$$

where $\psi(\cdot)$ is the Digamma function. Without loss of generality, let us consider the case where $\gamma = 0$ with no reparametrization, *i.e.*, $\boldsymbol{\theta}_t = \widetilde{\boldsymbol{\theta}}_t$. The results when $\gamma \neq 0$ and a mapping function $\Lambda(\cdot)$ for $\boldsymbol{\theta}_t$ is introduced are qualitatively the same. Clearly, what controls for the response to extreme observations in the conditional score $\boldsymbol{\nabla}_t(y_t, \boldsymbol{\theta}_t)$ is the degree of freedom parameter ν_t. When ν_t is small, say $\nu_t = 3$, the conditional distribution of y_t has a relatively higher probability mass in the tails, which means that extreme observations, which would be considered outliers under the conditionally Gaussian distribution, are more likely to be observed.

If we introduce the following mapping function for the unrestricted vector of parameter $\widetilde{\boldsymbol{\theta}}_t = (\widetilde{\mu}_t, \widetilde{\phi}_t, \widetilde{\nu}_t)^\top$:

$$\Lambda(\widetilde{\boldsymbol{\theta}}_t) \equiv \begin{cases} \mu_t \equiv & \widetilde{\mu}_t \\ \phi_t \equiv & \exp(\widetilde{\phi}_t) \\ \nu_t \equiv & \exp(\widetilde{\nu}_t) + c\,, \end{cases}$$

with $c = 2$ in order to ensure the existence of $V_{t-1}[y_t]$, then the GAS updating step for $\boldsymbol{\theta}_t$

[19] Note that the degrees of freedom parameter is assumed to be a real number larger than two, which makes the computation of the partial derivative straightforward.

when $\gamma = 0$ takes the form:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &\equiv \Lambda(\widetilde{\boldsymbol{\theta}}_{t+1}) \\ \widetilde{\boldsymbol{\theta}}_{t+1} &\equiv \boldsymbol{\kappa} + \mathbf{A}\mathcal{J}(\widetilde{\boldsymbol{\theta}}_t)^\top \boldsymbol{\nabla}_t(y_t, \boldsymbol{\theta}_t) + \mathbf{B}\widetilde{\boldsymbol{\theta}}_t \,, \end{aligned} \tag{9}$$

where $\boldsymbol{\kappa} \equiv (\kappa_\mu, \kappa_\phi, \kappa_\nu)^\top$, $\mathbf{A} \equiv \text{diag}(a_\mu, a_\phi, a_\nu)$ and $\mathbf{B} \equiv \text{diag}(b_\mu, b_\phi, b_\nu)$. In this particular case, the Jacobian matrix $\mathcal{J}(\widetilde{\boldsymbol{\theta}}_t)$ takes the form:

$$\mathcal{J}(\widetilde{\boldsymbol{\theta}}_t) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \exp(\widetilde{\phi}_t) & 0 \\ 0 & 0 & \exp(\widetilde{\nu}_t) \end{pmatrix} .$$

Constraints on the evolution of the GAS parameters can be easily considered by fixing the values of the $\mathbf{A}$ and $\mathbf{B}$ elements. For example, if the constraint $\nu_t = \nu$ has to be imposed, we set $a_\nu = b_\nu = 0$ during the (log-)likelihood maximization.

B. Mapping functions

Now we briefly discuss the choice of the mapping function $\Lambda(\cdot)$ for GAS models. We indicate the i–th element of $\boldsymbol{\theta}_t$ and $\widetilde{\boldsymbol{\theta}}_t$ as $\theta_{i,t}$ and $\widetilde{\theta}_{i,t}$, respectively. Analogously, we refer to the i–th element of the vector–valued mapping function $\Lambda(\cdot)$ as $\lambda_i(\cdot)$, such that $\lambda_i(\widetilde{\theta}_{i,t}) = \theta_{i,t}$.

Generally, there are three types of constraints we want to impose on $\theta_{i,t}$:

1) $\theta_{i,t} > c, \quad c \in \Re$

2) $\theta_{i,t} \in (a, b)$, for $a, b \in \Re$ and $b > a$

3) $\theta_{i,t} \in (a, b) \,|\boldsymbol{\theta}_t \in \Theta$ for $a, b \in \Re$ and $b > a$,

the additional case when $\theta_{i,t} \in \Re$, and thus $\widetilde{\theta}_{i,t} = \theta_{i,t}$, implicitly requires that $\lambda_i : \Re \to \Re$ is the identity function.

The first case, $\theta_{i,t} > c$, covers the situation where, for example, $\theta_{i,t}$ is a scale parameter and, consequently, its positiveness has to be imposed (*i.e.*, $c = 0$).[20] In this case, $\lambda_i : \Re \to [c, \infty)$, and the exponential link function, defined as:

$$\theta_{i,t} = \exp(\widetilde{\theta}_{i,t}) + c \,, \tag{10}$$

can be employed. The second case, $\theta_{i,t} \in (a, b)$, covers the situation where, for example, $p(\cdot; \boldsymbol{\theta}_t)$, is the asymmetric Student–t distribution of Zhu and Galbraith (2010), and $\theta_{i,t}$ is its skew parameter defined in $(0, 1)$. In the more general case we have $\lambda_i : \Re \to (a, b)$, and thus, the modified logistic function:

$$\theta_{i,t} = a + \frac{b - a}{1 + \exp(-\widetilde{\theta}_t)} \,, \tag{11}$$

can be employed. The last case, $\theta_{i,t} \in (a, b) \,|\boldsymbol{\theta}_t \in \Theta$, is more complicated and covers the situation where, for example, $p(\cdot; \boldsymbol{\theta}_t)$ is a multivariate Gaussian distribution and $\theta_{i,t}$ is one

[20] The case $\theta_{i,t} < c$ follows immediately.

element of its correlation matrix $\mathbf{R}_t$. Clearly, in this case $\theta_{i,t} \subseteq [-1,1]$, with the equivalence corresponding to the case $N = 2$. For the more general case $N > 1$, we need to ensure that $\mathbf{R}_t$ is positive definite, *i.e.*, $\mathbf{x}'\mathbf{R}_t\mathbf{x} > 0, \forall \mathbf{x} \in \Re^N$. Following Creal *et al.* (2011), we employ the hyperspherical coordinates transformation originally proposed by Pinheiro and Bates (1996) and subsequently discussed in Jaeckel and Rebonato (1999), Rapisarda *et al.* (2007) and Pourahmadi and Wang (2015). We define the general (h,k)–th lower diagonal element of $\mathbf{R}_t$ as $\rho_{hk,t} = \theta_{i,t}$ for $h > k$, $h < N$ and $\widetilde{\rho}_{hk,t} = \widetilde{\theta}_{i,t}$, for $i = 1, \ldots, N(N-1)/2$. Pourahmadi and Wang (2015) show that:

$$\rho_{hk,t} = c_{h1,t}c_{k1,t} + \sum_{m=2}^{h-1} c_{hm,t}c_{km,t} \prod_{l=1}^{m-1} s_{hl,t}s_{kl,t} + c_{hk,t} \prod_{l=1}^{h-1} s_{hl,t}s_{kl,1} \quad 1 \leq h < k \leq N\,,$$

where $c_{hk,t} \equiv \cos(\widetilde{\rho}_{hk,t})$ and $s_{hk,t} \equiv \sin(\widetilde{\rho}_{hk,t})$ for all $1 \leq h < k \leq N$ ensure that $\mathbf{R}_t \equiv \{\rho_{ij,t}\}_{i,j=1}^{N}$ is a proper correlation matrix.

These three specifications for $\lambda_i(\cdot)$ cover all the cases considered in this article and in the R package **GAS**. Additional information are reported in the R documentation. Fore details on $\Lambda(\cdot)$ and $\Lambda^{-1}(\cdot)$; see `help("UniMapParameters")` and `help("UniUnmapParameters")` in the univariate case and `help("MultiMapParameters")` and `help("MultiUnmapParameters")` in the multivariate case.

Affiliation:

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

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

Leopoldo Catania (corresponding author)
Department of Economics and Finance
Faculty of Economics
University of Rome, "Tor Vergata"
Via Columbia, 2
00133 Rome, Italy
E-mail: leopoldo.catania@uniroma2.it