Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Meirelles Bodin de Moraes

ScoreDrivenModels.jl: a Julia Package for Generalized Autoregressive Score Models

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor : Prof. Alexandre Street de Aguiar
Co-Advisor: Prof. Cristiano Augusto Coelho Fernandes

Rio de Janeiro
April 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme Meirelles Bodin de Moraes

ScoreDrivenModels.jl: a Julia Package for Generalized Autoregressive Score Models

Dissertation presented to the Programa de Pós-Graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the Examination Committee.

Prof. Alexandre Street de Aguiar
Advisor
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Cristiano Augusto Coelho Fernandes
Co-Advisor
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Fernando Luiz Cyrino Oliveira
Departamento de Engenharia Industrial – PUC-Rio

Prof. Rutger Lit
Vrije Universiteit Amsterdam – VU Amsterdam

Rio de Janeiro, April 13th, 2021

Guilherme Meirelles Bodin de Moraes

Guilherme Meirelles Bodin de Moraes received his B.Sc. degree in Electrical Engineering in 2018 from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil and his M.Sc. degree in Generalistic Engineering in 2018 from École Centrale de Marseille, France. During his undergraduate program, he spent two years as a double degree student at the École Centrale de Marseille, and was also an intern at Visagio in Rio de Janeiro. Since 2017, he has been actively participating in research projects at the Laboratory of Applied Mathematical Programming and Statistics (LAMPS) in the Department of Electrical Engineering at PUC-Rio.

Bibliographic data

Moraes, Guilherme Meirelles Bodin de

ScoreDrivenModels.jl: a Julia Package for Generalized Autoregressive Score Models / Guilherme Meirelles Bodin de Moraes; advisor: Alexandre Street de Aguiar; co-orientador: Cristiano Augusto Coelho Fernandes. – 2021.

51 f. : il. color. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, Rio de Janeiro, 2021.

Inclui bibliografia.

1. Engenharia Elétrica – Teses. 2. Modelos orientados por score. 3. Modelos generalizados de score autorregressivos. 4. Modelos de séries temporais. 5. Parâmetros variantes no tempo. 6. Modelos não-Gaussianos. I. Aguiar, Alexandre Street de. II. Coelho Fernandes, Cristiano Augusto. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. ScoreDrivenModels.jl: Generalized Autoregressive ScoreModels in Julia .

CDD: 621.3

Acknowledgments

I would like to thank my parents, Angela and Paulo, for their attention and uplifting attitude.

I would like to thank my girlfriend, Bianca, for the love and support.

I would like to thank my advisor Alexandre Street for the excellent orientation on all personal, professional, and academic backgrounds. I learned many valuable lessons during the orientation. I also extend my thanks to my co-advisor Cristiano Fernandes for his attentive inputs throughout the development of my research and for his remarkable courses on the many subjects of time-series.

I wish to thank my colleagues and professors at LAMPS for the outstanding environment and the daily conversations.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Meirelles Bodin de Moraes, Guilherme; Street de Aguiar, Alexandre (Advisor); Coelho Fernandes, Cristiano Augusto (Co-Advisor). **ScoreDrivenModels.jl: a Julia Package for Generalized Autoregressive Score Models**. Rio de Janeiro, 2021. 51p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Score-driven models, also known as generalized autoregressive score (GAS) models, represent a class of observation-driven time series models. They possess desirable properties for time series modeling, such as the ability to model different conditional distributions and to consider time-varying parameters within a flexible framework. In this dissertation, we present ScoreDrivenModels.jl, an open-source Julia package for modeling, forecasting, and simulating time series using the framework of score-driven models. The package is flexible with respect to model definition, allowing the user to specify the lag structure and which parameters are time-varying or constant. It is also possible to consider several distributions, including Beta, Exponential, Gamma, Lognormal, Normal, Poisson, Student's t, and Weibull. The provided interface is flexible, allowing interested users to implement any desired distribution and parametrization.

Keywords

score-driven models; generalized autoregressive score models; time series models; time-varying parameters; non-Gaussian models

Resumo

Meirelles Bodin de Moraes, Guilherme; Street de Aguiar, Alexandre; Coelho Fernandes, Cristiano Augusto. **ScoreDrivenModels.jl: Pacote em Julia para Modelos Generalizados Autorregressivos com Score**. Rio de Janeiro, 2021. 51p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Os modelos orientados por score, também conhecidos como modelos generalizados de score autorregressivo (GAS), representam uma classe de modelos de séries temporais orientados por observação. Eles possuem propriedades desejáveis para modelagem de séries temporais, como a capacidade de modelar diferentes distribuições condicionais e considerar parâmetros variantes no tempo dentro de uma estrutura flexível. Neste trabalho, apresentamos ScoreDrivenModels.jl, um pacote Julia de código aberto para modelagem, previsão e simulação de séries temporais usando a estrutura de modelos baseados em score. O pacote é flexível no que diz respeito à definição do modelo, permitindo ao usuário especificar a estrutura de atraso e quais parâmetros são variantes no tempo ou constantes. Também é possível considerar várias distribuições, incluindo Beta, Exponencial, Gama, Lognormal, Normal, Poisson, Student's t e Weibull. A interface fornecida é flexível, permitindo aos usuários interessados implementar qualquer distribuição e parametrização desejada.

Palavras-chave

Modelos orientados por score; modelos generalizados de score autorregressivo; modelos de séries temporais; parâmetros variantes no tempo; modelos não-Gaussianos

Table of contents

List of figures

List of tables

1
Introduction

Time series models with time-varying parameters have become increasingly popular over the years due to their advantages in capturing dynamics of series of interest. According to [1], the mechanism driving parameter dynamics in this general class of models can be of two types: parameter-driven, as in state space models [2–4], or observation-driven. In this work, we will focus on a recently proposed class of observation-driven models wherein the score of the predictive density is used as the driver for parameter updating [5, 6]. These models have been referred to as generalized autoregressive score (GAS) models, dynamic conditional score models, or simply score-driven models. Additionally, it has been demonstrated that well-established observation driven models, such as the GARCH [7] and conditional duration models [8], are particular cases of the score-driven framework.

One of the main advantages of the GAS framework is its flexibility, as it is possible to consider different non-Gaussian distributions. Moreover, the updating mechanism is intuitive and determined by the score of the chosen distribution. These properties have led GAS models to be applied in numerous fields, such as finance [9, 10], actuaries [11, 12], risk analysis [13, 14], and renewable generation [15, 16]. We also refer the interested reader to a large online repository of works on GAS models at `http://www.gasmodel.com`.

This wide range of applications has motivated the development of software packages for this class of models. For instance, there are open-source packages in Python [17], R [18], and, recently, the data consultancy company Nlitn have made publicly available Time Series Lab (`https://timeserieslab.com`), a free software developed by some of the authors of the theory developed in [5, 6].

In this work, we present a novel open-source GAS package fully implemented in Julia [19] named ScoreDrivenModels.jl [20]. One of Julia's main advantages is to avoid the so-called two-language problem, i.e., the dependence on subroutines implemented in lower-level languages such as C, C++, or Fortran. Julia achieves this by providing a high-level programming syntax that allows for rapid prototyping and development without sacrificing computational performance. Thus, by providing an open-source package completely

written in Julia, we facilitate development and contributions by users while also maintaining a high level of code transparency. The package allows users to specify a wide variety of GAS models by choosing the conditional distribution, the autoregressive structure, and which parameters are time-varying. Finally, initialization procedures are implemented to make the estimation process more robust for the case of seasonal time series.

The main contributions of this work are:

1. We extend the existing scientific literature [18], which presented a GAS$(1, 1)$ implementation, by providing an open-source implementation of the procedures for GAS(p, q) models in an open-source framework.

2. The proposed implementation is entirely programmed in Julia avoiding the so-called two language problem in the related literature [18].

3. We extend the existing literature [18] by providing relevant new features and a correction. Firstly, mistaken calculations of the inverse scaling and inverse square root scaling present in [18] are corrected. Secondly, we devise a heuristic for estimating the initial parameters which makes the overall estimation process more robust. Finally, we implement quantile residuals as an additional feature to be used in model diagnostics.

The remainder of this dissertation is organized as follows. Section 2 provides a brief overview of the GAS framework. In Section 3, the ScoreDrivenModels.jl package is presented, including the model specification, estimation, forecasting, and simulation. Section 4 presents examples of applications to illustrate the use of the package. Conclusions are drawn in Section 5. Finally, the Appendix provides the derivation of the score for each implemented distribution.

2
Score Driven Models

Let $y_t \in \mathbb{Y} \subseteq \mathbb{R}$ denote the dependent variable of interest, $f_t \in \mathcal{P} \subset \mathbb{R}^k$ be a vector of time-varying parameters, and $F^t = \{f_0, f_1, \ldots, f_t\}$ and $Y^t = \{y_1, \ldots, y_t\}$. For simplicity of notation we define $\mathcal{F}_{t-1} = (F^{t-1}, Y^{t-1})$ denote the sets of available information until time t. We assume that y_t is generated by the probability density function conditioned on the available information (past data and time-varying parameters) and on the hyperparameter vector θ, which contains the constant parameters. The predictive distribution of y_t has a closed form, represented as:

$$p(y_t|f_t, \mathcal{F}_{t-1}; \theta) \tag{2-1}$$

In score-driven models, we start by choosing the updating mechanism for the time-varying parameters f_t is given by the following equation, referred to as a GAS(p, q) mechanism:

$$f_{t+1} = \omega + \sum_{i=1}^{p} A_i s_{t-i+1} + \sum_{j=1}^{q} B_j f_{t-j+1}, \tag{2-2}$$

where ω is a vector of constants, coefficient matrices A_i and B_j have appropriate dimensions for $i = 1, \ldots, p$ and $j = 1, \ldots, q$, and $s_t = s_t(y_t, f_t, F_{t-1}, Y_{t-1}; \theta)$ is an appropriate function of past data. The unknown coefficients in Eq. (2-2) are functions of the vector of hyperparameters θ; that is, $\omega = \omega(\theta)$, $A_i = A_i(\theta)$, and $B_j = B_j(\theta)$. At instant t, the update of the time-varying f_t for the next period $t+1$ is conducted through Eq. (2-2), with

$$s_t = \mathcal{I}_{t|t-1}^{-d} \cdot \nabla_t, \qquad \nabla_t = \frac{\partial \ln p(y_t|f_t, \mathcal{F}_{t-1}; \theta)}{\partial f_t}, \tag{2-3}$$

where ∇_t is the called the score and $\mathcal{I}_{t|t-1}^{-d}$ is the scaled Fisher information of the probability density $p(y_t|f_t, \mathcal{F}_{t-1}; \theta)$. The scaling coefficient d commonly takes values in $\{0, \frac{1}{2}, 1\}$. It is worth mentioning that in the case where $d = \frac{1}{2}$, it follows that $\mathcal{I}_{t|t-1}^{-\frac{1}{2}}$ results from the Cholesky decomposition of $\mathcal{I}_{t|t-1}^{-1}$.

As a consequence of the time-varying mechanism for the distribution parameters presented in Eq. (2-2), the conditional distribution of a GAS model is capable of continuously changing based on the considered data. This property is illustrated in Fig. 2.1. For instance, if the time series contains

occasional volatility spikes, the model can capture this behavior through the time-varying nature of the parameters.

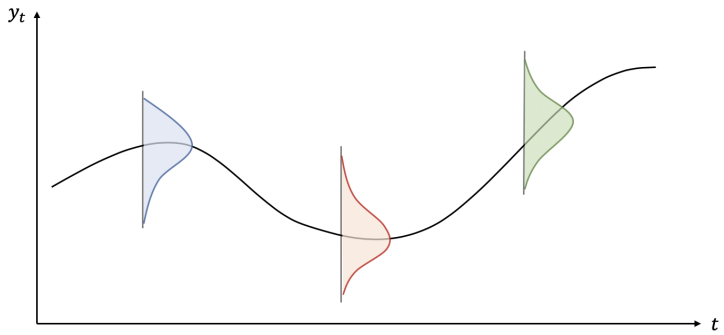

Figure 2.1: The GAS framework allows the conditional distribution to continuously change based on the data.

2.1 Parametrization

In the GAS updating mechanism (2-2), the parameter $f_t \in \mathcal{P} \subset \mathbb{R}^k$ is sometimes bounded – for example, in a Gaussian GAS model where the variance is time-varying, we would have $f_t = \sigma_t^2$ which can only assume positive values by definition. However, in some cases the recursion can lead to updates of $f_t \notin \mathcal{P}$. A solution is to reparametrize the equations in order to guarantee $f_t \in \mathcal{P}$ for every update. To that end, we follow the procedure described in [5] and present an example to illustrate it.

Let f_t be the vector of time-varying parameters of a Gaussian distribution. From the properties of the distribution, it follows that $\mu_t \in \mathbb{R}$ and $\sigma_t^2 \in \mathbb{R}^+$. Let us define a new time-varying parameter $\tilde{f}_t \in \mathbb{R}^k$ and a map $h : \mathcal{P} \to \mathbb{R}^k$, which we denote the `link` function in the package. In the case of the Gaussian distribution, a useful approach is to have an `IdentityLink` for μ_t and a `LogLink` for σ_t^2 as follows:

$$f_t = \begin{bmatrix} \mu_t \\ \sigma_t^2 \end{bmatrix}, \tilde{f}_t = \begin{bmatrix} \mu_t \\ \ln \sigma_t^2 \end{bmatrix} \tag{2-4}$$

Note that the use of this parametrization will affect the recursion in Eq. (2-2) as well as the final expressions of s_t. Thus, let us derive the new recursion for the (2-2), but this time with the guarantee that every update of f_t respects $f_t \in \mathcal{P}$. To that end, we also define the inverse map $h^{-1}(\cdot)$ denoted in the software as the `unlink` function: $f_t = h^{-1}(\tilde{f}_t)$. Then, we can define the GAS updating recursion utilizing the parametrization:

$$\tilde{f}_{t+1} = \omega + \sum_{i=1}^{p} A_i \tilde{s}_{t-i+1} + \sum_{j=1}^{q} B_j \tilde{f}_{t-j+1} \tag{2-5}$$

The unknown coefficients in (2-5) remain as functions of θ; that is, $\omega = \omega(\theta)$, $A_i = A_i(\theta)$, and $B_j = B_j(\theta)$. However, this time, we update the linked version of the time-varying parameters $\tilde{f}_t$ using (2-5). It is important to note that the expressions of $\tilde{s}_t$ are different for each scaling $d \in \{0, \frac{1}{2}, 1\}$. To compute these, we must use the derivative $\dot{h}$ of the map h, which is simply its Jacobian. In our example, h is defined as

$$h\left(\begin{bmatrix}\mu_t \\ \sigma_t^2\end{bmatrix}\right) = \begin{bmatrix}h_1(\mu_t) \\ h_2(\sigma_t^2)\end{bmatrix} = \begin{bmatrix}\tilde{\mu}_t \\ \tilde{\sigma}_t^2\end{bmatrix} = \begin{bmatrix}\mu_t \\ \ln \sigma_t^2\end{bmatrix} \tag{2-6}$$

and its Jacobian is defined as

$$\dot{h}\left(\begin{bmatrix}\mu_t \\ \sigma_t^2\end{bmatrix}\right) = \begin{bmatrix}\frac{\partial h_1(\mu_t)}{\partial \mu_t} & \frac{\partial h_1(\mu_t)}{\partial \sigma_t^2} \\ \frac{\partial h_2(\sigma_t^2)}{\partial \mu_t} & \frac{\partial h_2(\sigma_t^2)}{\partial \sigma_t^2}\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & \frac{1}{\sigma_t^2}\end{bmatrix} \tag{2-7}$$

Note that $\dot{h}$ is always a diagonal matrix. The reparametrized score derivations for different scalings and different types of maps are presented in Appendix A. Given the following definitions

$$\dot{h}(f_t) = \left.\frac{\partial h(f_t)}{\partial \tilde{f}_t}\right|_{f_t}, \qquad \mathcal{J}_{t|t-1}\mathcal{J}_{t|t-1}^{\top} = \mathcal{I}_{t|t-1}^{-1}, \qquad \tilde{\nabla}_t = \left(\dot{h}\right)^{-1} \nabla_t, \tag{2-8}$$

then the linked scaled score $\tilde{s}_t$ can be computed as follows:

$$\tilde{s}_t = \left(\dot{h}\right)^{-1} \nabla_t, \text{ for } d = 0, \tag{2-9}$$

$$\tilde{s}_t = \mathcal{J}_{t|t-1} \nabla_t, \text{ for } d = \frac{1}{2}, \tag{2-10}$$

$$\tilde{s}_t = \dot{h} \mathcal{I}_{t|t-1}^{-1} \nabla_t, \text{ for } d = 1. \tag{2-11}$$

It should be noted that in the work of [18] there is an imprecision that leads to significant errors when estimating models with inverse square root scaling ($d = \frac{1}{2}$). An example that shows the effects of this mistake is shown in Appendix A.4.

2.2 Maximum likelihood estimation

In score-driven models the vector of hyperparameters θ is estimated via maximum likelihood:

$$\hat{\theta} = \arg\max_{\theta} \sum_{t=1}^{N} \ln p(y_t | f_t, \mathcal{F}_{t-1}; \theta) \tag{2-12}$$

Evaluating the log-likelihood function of a GAS model is particularly simple. Given values for the constant parameters θ, the GAS updating equation (2-2) outputs the conditional distribution at each time period, which generally has a closed form. Thus, it suffices to look at $\ln p(y_t|f_t, \mathcal{F}_{t-1}; \theta)$ for a particular value of θ.

The maximum likelihood estimation is performed by global optimization methods such as L-BFGS [21] and Nelder-Mead [22]. If the optimization routine used to find the optimum requires derivatives this should be done numerically. Constrained optimization can also be applied, using, for instance, a Newton interior points method.

2.3 Forecasting

Forecasting and simulation of future scenarios are among the main goals in time series analysis. In [23], details of the procedure for out-of-sample confidence intervals for the time-varying parameters are discussed. The procedure discussed in Section 4.1 of [23] is currently implemented in **ScoreDrivenModels.jl** as follows:

1. Given $\hat{\theta}_T$ and the filtered state $\hat{f}_{T+1}$, draw S values $y^1_{T+1}, \ldots, y^S_{T+1}$ from the estimated conditional density at $T+1$: $y_{T+1} \sim p(y_{T+1}|\hat{f}_{T+1}, \hat{\theta}_T)$ for $s = 1, \ldots, S$.

2. Use $y^1_{T+1}, \ldots, y^S_{T+1}$ and the recursion (2-2) to obtain the filtered values $\hat{f}^1_{T+2}, \ldots, \hat{f}^S_{T+2}$.

3. Repeat steps 1 and 2 H times for H steps ahead generating one new value of y and f per scenario s.

Once the procedure is over, S scenarios for the observations within the entire horizon, y^s_{T+k} for $k = 1, \ldots, H$ and $s = 1, \ldots, S$ have been simulated. Based on these set of scenario, one can calculate quantile forecasts, build empirical distributions, or use them to feed decision under uncertainty models, such as stochastic programming. Note that this method solely considers the uncertainty of innovations. The consideration of uncertainty on both innovations and parameters, as discussed in [23], is considered future work for the package.

2.4 Diagnostics

One way to diagnose if score-driven models has captured the intrinsic structure of the time series and its distribution is to analyze its quantile residuals [24]. Under correct model specification the PIT, p_t, of the residuals is a standard uniform random variable $\mathcal{U}(0,1)$. If this is the case, the quantile residuals, as given by equation 2-13 where Φ is the cumulative standard Gaussian distribution, are normally distributed by construction and one can diagnose the model residuals using standard statistical tests. The quantile residuals are calculated by evaluating quantiles of the cumulative density function of the PIT of residuals.

$$r_{\text{quantile}} = \Phi^{-1}(p_t) \tag{2-13}$$

In the case of models with a discrete distribution the PIT of the residuals is not a standard uniform random variable $\mathcal{U}(0,1)$ [25]. To address this issue the randomized PIT is estimated. We draw each p_t at random from the following uniform distribution

$$p_t \sim \mathcal{U}\left(F_t(y_t - 1), F_t(y_t)\right) \tag{2-14}$$

Then p_t are again $\mathcal{U}(0,1)$ as their distribution is a mixture of uniform distributions on sub-intervals in $[0,1]$ and one can diagnose the residuals using standard statistical tests.

2.5 Checking probabilistic forecasts

Even if a model is estimated under correct specification, it still provides no guarantee of good out-of-sample forecasts. When forecasting is a goal, techniques to benchmark and track the performance of the forecasts should be implemented. In particular, to evaluate probabilistic forecasts one should keep track of the forecasting errors and scores such as the Continuous Ranked Probability Score (CRPS) [26]. The CRPS is calculated using the a predictive density $\hat{F}_{t+k|t}$ and the observation y_{t+k}:

$$CRPS(\hat{F}_{t+k|t}, y_{t+k})_{t,k} = \int_x \left[\hat{F}_{t+k|t}\left(x\right) - 1\left(y_{t+k} \leq x\right)\right]^2 \mathrm{d}x \tag{2-15}$$

It is very common that the probabilistic forecast relies on simulated scenarios $(X_i, \ldots, X_m \sim \hat{F}_{t+k|t})$ of that predictive density. Because of this we can approximate the cumulative distribution function (cdf) by the equation

$$\hat{F}_{t+k|t}(x) = \frac{1}{m}\sum_{i=1}^{m} 1\{X_i \leq x\} \tag{2-16}$$

With the approximated cdf the CRPS reduces to

$$CRPS(\hat{F}_{t+k|t}, y_{t+k})_{t,k} = \frac{1}{m}\sum_{i=1}^{m}|X_i - y_{t+k}| + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}|X_i - X_j| \qquad (2\text{-}17)$$

Notice here that if one makes a probabilistic forecast where all of the scenarios are the same, equivalent to a deterministic forecast, the CRPS is equivalent to the Mean Absolute Error (MAE) as the contribution for one lead time will be exactly $|\bar{X} - y|$. The implementation of equation 2-17 is computationally inefficient $\mathcal{O}(m^2)$ [27] and can be improved by adopting order statistics $X_{(1)}, \ldots, X_{(m)}$, thus achieving an average $\mathcal{O}(m \log m)$ performance.

$$CRPS(\hat{F}_{t+k|t}, y_{t+k})_{t,k} = \frac{2}{m^2}\sum_{i=1}^{m}(X_{(i)} - y_{t+k})(m1\{y_{t+k} < X_{(i)}\} - i + \frac{1}{2}) \quad (2\text{-}18)$$

Equation 2-18 represents the CRPS of a forecast made at time t and lead time k. To evaluate the probabilistic forecasts performance one should gather the CRPS in different out-of-sample periods and average the calculated CRPS for each lead time. The preferred model is the one with minimum CRPS.

$$CRPS(\hat{F}_{t+k|t}, y_{t+k})_{k} = \frac{1}{T}\sum_{t=1}^{T} CRPS(\hat{F}_{t+k|t}, y_{t+k})_{t,k} \qquad (2\text{-}19)$$

3
The ScoreDrivenModels.jl package

ScoreDrivenModels.jl enables users to create and estimate score-driven models and to perform forecasting and simulation while working purely in Julia. Its API allows users to choose between different distributions, scaling values, lag structures, and optimization methods. The basic code structure allows contributors to add new distributions and optimization methods; technical details about adding new features are available in the package documentation. Installation of the package is easily conducted using the Julia Package manager:

```
pkg> add ScoreDrivenModels
```


3.1
Model specification

To create a `Model`, the user must specify 1) the desired distribution, 2) the scaling, 3) the lag structure, and 4) which parameters should be considered time-varying.

1. The lag structure in a GAS(p, q) model can be specified in two ways: either through integers `p` and `q`, which results in all lags from `1` to `p` and `1` to `q` being added, or through arrays of integers `ps` and `qs` containing only the desired lags.

2. To specify the distribution, the user needs to choose a distribution among the available ones that have an interface with **Distributions.jl**. The list of available distributions is displayed in Table 3.1. Furthermore, we refer the interested reader to Appendix B, where we provide details on the score calculations for each probability density made available in the package.

3. The scaling is specified by defining the value of d, which can be 0, 1, or $\frac{1}{2}$, respectively the identity scaling, inverse scaling, and inverse square-root scaling.

4. In order to define which distribution parameters should be time-varying, the keyword argument `time_varying_params` can be used. Note that the default behavior is to have all parameters as time-varying.

Distribution	Number of parameters	Identity scaling	Inverse scaling	Inverse square-root scaling
Beta	2	✓	✓	✓
BetaFourParameters	4	✓	–	–
Exponential	1	✓	✓	✓
Gamma	2	✓	✓	✓
LogitNormal	2	✓	✓	✓
LogNormal	2	✓	✓	✓
NegativeBinomial	2	✓	–	–
Normal	2	✓	✓	✓
Poisson	1	✓	✓	✓
TDist	1	✓	✓	✓
TDistLocationScale	3	✓	✓	✓
Weibull	2	✓	–	–

Table 3.1: List of currently implemented distributions and scalings. # represent the number of parameters of the distribution.

Once the model is specified, the unknown parameters that must be estimated are automatically represented as `NaN` within the `Model` structure. As an example, a GAS$(1, 2)$ model with lognormal distribution and inverse square-root scaling can be created by writing the following line of code:

```
julia> Model(1, 2, LogNormal, 0.5)

Model{LogNormal,Float64}([NaN, NaN], Dict(1=>[NaN 0.0; 0.0 NaN]),
    Dict(2=>[NaN 0.0; 0.0 NaN],1=>[NaN 0.0; 0.0 NaN]), 0.5)
```

`Dict` is the Julia data structure for dictionaries. Its use allows code flexibility enabling computational simplifications for complex lag structures. As displayed above, the unknown constant parameters to be estimated are set as `NaN`. In this case, the constant parameters considered in vector ω are A_1, B_1, and B_2.

In some applications, however, the user might define only one of the distribution parameters as time-varying. In the example below, the only time-varying parameter is μ_t, so the keyword argument `time_varying_params` indicates a vector with only one element, `[1]`, representing the first parameter of the lognormal distribution. A table that indicates the distribution parameters and their orders is available in the package documentation. The choice of the time-varying parameter can be expressed by the following code:

```
julia> Model(1, 2, LogNormal, 0.5; time_varying_params = [1])

Model{LogNormal,Float64}([NaN, NaN], Dict(1=>[NaN 0.0; 0.0 0.0]),
    Dict(2=>[NaN 0.0; 0.0 0.0],1=>[NaN 0.0; 0.0 0.0]), 0.5)
```

Users can also specify the lag structure by passing only the lags of interest. Note that this feature is equivalent to defining that matrices A_i and B_j are equal to zero for certain values i and j. An example is a model that uses lags 1 and 12, which means that only the matrices A_1, A_{12}, B_1, and B_{12} have nonzero entries:

```
julia> Model([1, 12], [1, 12], LogNormal, 0.5)

Model{Normal,Float64}([NaN, NaN],
    Dict(12=>[NaN 0.0; 0.0 NaN],1=>[NaN 0.0; 0.0 NaN]),
    Dict(12=>[NaN 0.0; 0.0 NaN],1=>[NaN 0.0; 0.0 NaN]), 0.5)
```

3.2 Estimation

Once the model is specified, the next step is estimation. Users can choose from different optimization methods provided by **Optim.jl** [28]. Since this optimization problem is non-convex, there is no guarantee that the optimal value found by the optimization method is the global optimum. To increase the chances of finding the global optimum, we run the optimization algorithm for different initial parameter values. The default method is Nelder-Mead with 3 random initial parameter values, but the optimization interface is highly flexible. Users can customize convergence tolerances, choose initial parameter values, and, depending on the optimization method, choose bounds for the parameters. By default, these initial values are the unconditional mean of f_{t+1} which is given by

$$\mathsf{E}[f_{t+1}] = \omega\left(I - \sum_{j=1}^{q} B_j\right)^{-1}. \tag{3-1}$$

As an illustration, let us estimate a GAS model using the same data and specification used in the R package **GAS** [18] paper with the function `fit!`, the data is also available in package repository [20]. The data represents the monthly US inflation measured as the logarithmic change of the consumer price index. The model can be estimated as follows:

```
julia> Random.seed!(123)
julia> y = vec(readdlm("../test/data/cpichg.csv"))
julia> gas = Model(1, 1, TDistLocationScale, 0.0,
                   time_varying_params=[1, 2])
julia> f = fit!(gas, y)

Round 1 of 3 - Log-likelihood: -178.20653071457616
Round 2 of 3 - Log-likelihood: -185.28141597363677
Round 3 of 3 - Log-likelihood: -178.20684252730365
```

Users also have the option to check more detailed results of the optimization procedure by changing the keyword argument `verbose`. The default value of this argument is 1; to check the optimization summary, users should set the verbose level to 2, and to see the value of the objective function at each iteration of the optimization, it should be set to 3. To avoid the printing of outputs, users can set `verbose = 0`. To illustrate, results with level 2 is depicted below.

```
julia> gas = Model(1, 1, TDistLocationScale, 0.0,
                   time_varying_params=[1, 2])
julia> f = fit!(gas, y; verbose=2)

Round 1 of 3 - Log-likelihood: -178.20685880563667
Round 2 of 3 - Log-likelihood: -178.2067301284657
Round 3 of 3 - Log-likelihood: -178.20686956876364

Best optimization result:
 * Status: success

 * Candidate solution
    Minimizer: [3.78e-02, -2.58e-01, 1.87e+00,  ...]
    Minimum:   6.456766e-01

 * Found with
    Algorithm:     Nelder-Mead
    Initial Point: [3.33e-01, 5.64e-01, 2.88e-01,  ...]

 * Convergence measures
    standard-deviation <= 1.0e-06

* Work counters
    Seconds run:   0  (vs limit 100000000)
    Iterations:    513
    f(x) calls:    802
```

As mentioned before, while the maximization of the log-likelihood is done by default through the Nelder-Mead method with 3 random initial values, these features can be changed by the user. For example, to use the L-BFGS algorithm with 5 random initial values:

```
julia> gas = Model(1, 1, TDistLocationScale, 0.0,
                   time_varying_params=[1, 2])
```

```
julia> f = fit!(gas, y; opt_method=LBFGS(gas, 5))
Round 1 of 5 - Log-likelihood: -178.20649316732568
Round 2 of 5 - Log-likelihood: -178.20649425266643
Round 3 of 5 - Log-likelihood: -178.20649317466822
Round 4 of 5 - Log-likelihood: -189.08548581293175
Round 5 of 5 - Log-likelihood: -355.33018234566873
```

Once the estimation step is finished, the user can query the results by calling the function `results`:

```
julia> results(f)

------------------------------------------------------
Distribution:                 Distributions.LocationScale{
                                        Float64,TDist{Float64}}
Number of observations:       276
Number of unknown parameters: 7
Log-likelihood:               -178.2065
AIC:                          370.4130
BIC:                          395.7558
------------------------------------------------------
Parameter      Estimate   Std.Error     t stat   p-value
omega_1          0.0374      0.0311     1.2016    0.2686
omega_2         -0.2599      0.1409    -1.8454    0.1075
omega_3          1.8758      0.2914     6.4380    0.0004
A_1_11           0.0717      0.0184     3.8884    0.0060
A_1_22           0.4538      0.2139     2.1216    0.0715
B_1_11           0.9432      0.0272    34.6438    0.0000
B_1_22           0.8556      0.0743    11.5141    0.0000
```

This result matches the example discussed in [18] with the exception of `omega_3`, due to a difference in parametrization between the two packages. Once the parameter is recovered to its original parametrization, the result becomes the same.

3.3 Residuals analysis

After estimating score-driven models one can use the resulting residuals for model diagnostics. With this aim in mind, auto-correlation function, histogram and QQ plot can be obtained using the following code:

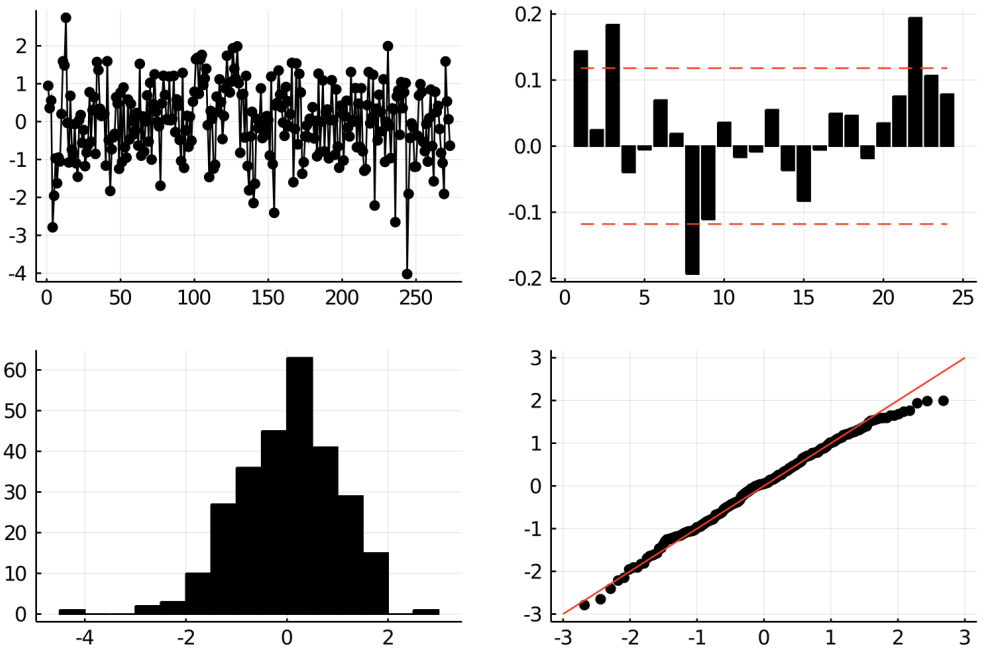

Figure 3.1: Plots to diagnose the quantile residuals of the model. Top left - quantile residuals, top right - autocorrelation function, bottom left - histogram, bottom right - QQ plot.

```
julia> plot(f)
```

In this case the output indicated in Figure 3.1 that there is still some remaining structure in lags 1, 3, 8 and 22 and an outlier near the observation 250.

3.4 Forecasting and simulation

Forecasting in this framework is done by simulation as per Section 2.3. Function `forecast` runs the procedure proposed by [23] and returns a `Forecast` structure that has the expected value for time-varying parameters, observations, and the related scenarios used to find them. By default the structure also stores the 2.5%, 50% and 97.5% quantiles.

Next, we will present forecasting results using the previously estimated US inflation data. In the example below, the first column is the location parameter, the second column is the scale parameter, and the third column represents the degrees of freedom parameter.

```
julia> forec = forecast(y, gas, 12)
julia> forec.parameter_forecast

12x3 Array{Float64,2}:
0.134315  0.159539  6.52619
```

```
 0.16581   0.166047  6.52619
 0.196429  0.171559  6.52619
 0.219776  0.175266  6.52619
 0.24502   0.17878   6.52619
 0.267964  0.179804  6.52619
 0.288719  0.182248  6.52619
 0.308389  0.184335  6.52619
 0.327624  0.185162  6.52619
 0.348591  0.18611   6.52619
 0.363832  0.186892  6.52619
 0.380106  0.187219  6.52619
```

We can also obtain the scenarios of observations, y^s_{T+k} for $k = 1, \ldots, H$ and $s = 1, \ldots, S$, that generated the above forecasted values as follows:

```
julia> forec.observation_scenarios
```

```
12x10000 Array{Float64,2}:
 1.3839    -0.378146   ...   0.561549   0.0415801
 0.971144   0.116446         0.478346   0.290846
 1.1637     0.439987         0.157433  -0.195271
 0.50442   -0.0417178        0.204036  -0.549865
 1.39936    0.149797         0.606875  -0.263888
 1.04027    0.564609   ...  -0.234421   0.139999
 0.609759   1.2166           0.236292   0.408869
 0.693315   0.057054         0.802398   0.285412
 0.908195   1.06543          1.22845    0.256618
 1.89531    0.566139         0.808426   0.121839
 1.7671     0.813116   ...   0.639728   0.0408979
 0.840164   0.72076          0.316973   0.300578
```

For the sake of clarity, the forecast $\hat{y}_{T+k}$ is the mean of the distribution with parameters `forec.parameter_forecast`.

3.5 Cross validation

One way to check the model forecasting performance is to estimate and forecast the same model multiple times in a rolling window scheme. In a simple manner one define a fixed number of steps ahead to predict and perform multiple estimations and forecasts for different periods of time. **ScoreDrivenModels.jl** has a `cross_validation` function that only requires the

user to define a model, the number of steps ahead to forecast and an index of the time series where the user wants to begin the rolling window. The function will perform a rolling window scheme of estimations and forecasts for all the periods from the desired index until the last period where it is possible to compare forecasts and observations. Let us show an example where we want to test the forecasting performance of two Student's t models and plot the mean absolute errors and mean CRPSs per lead time.

```
julia> gas_t = Model(1, 1, TDistLocationScale, 1.0;
                     time_varying_params = [1])
julia> gas_t_1_2 = Model(1, 2, TDistLocationScale, 1.0;
                     time_varying_params = [1])
julia> steps_ahead = 50
julia> first_idx = 150
julia> b_t = cross_validation(gas_t, y, steps_ahead, first_idx)
julia> b_t_1_2 = cross_validation(gas_t_1_2, y, steps_ahead, first_idx)
julia> plot(b_t, "GAS(1, 1) Student t")
julia> plot!(b_t_1_2, "GAS(1, 2) Student t"; legend = :topleft)
```


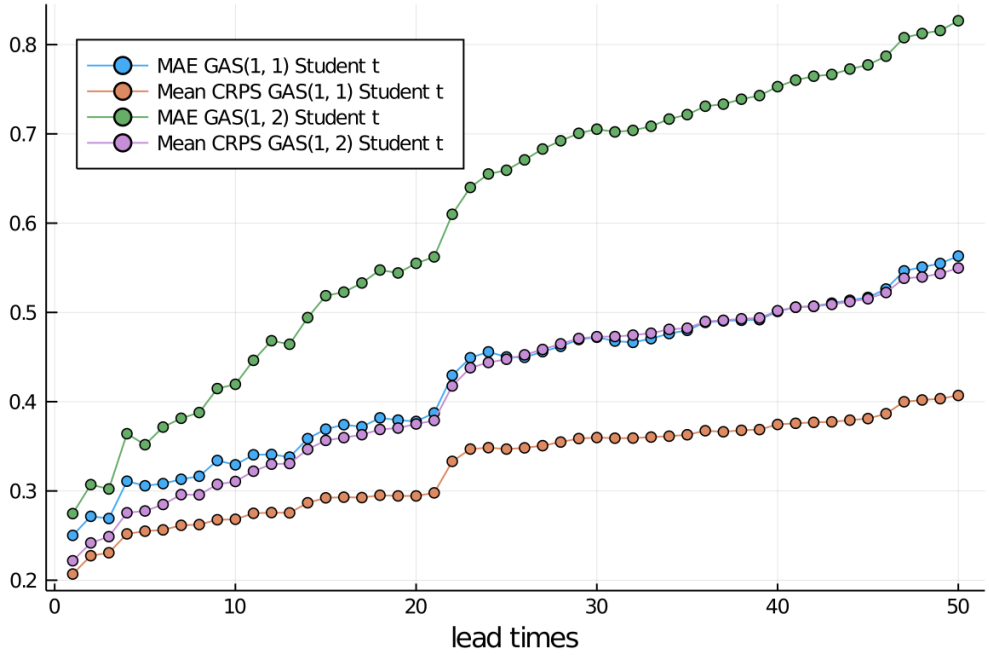

Figure 3.2: MAE and mean CRPS per lead time of two models.

Using this feature, it is straightforward to compare the forecasting performance of different models. In the example above we compare a GAS$(1,1)$ and a GAS$(1,2)$ models with inverse scaling and time-varying location parameter. From Fig. 3.2, it can be seen that the GAS$(1,1)$ model presented

superior performance in terms of mean CRPS and MAE in comparison to the GAS$(1, 2)$.

4
Applications

4.1
Hydropower generation in Brazil

The Brazilian system operator regularly publishes an indicator of how much energy can be generated from water inflows among all hydropower plants in the country. This indicator is referred to as Natural Inflow Energy (NIE). Usually, NIE is computed in daily basis per region and then aggregated in a monthly basis as the average of each month. The NIE works as an indicator of the health of the reservoir levels in the Brazilian electric system and it is an important variable for the definition of energy spot prices in each Brazilian sub-system.

Due to the high dependency of water resources, in Brazil, the monthly NIE is a key component of system operations and infrastructure planning studies. In most decision under uncertainty methodologies, it is essential to have simulated scenarios describing the empirical distribution rather than simple point forecasts.

In this section, we present an example that generates scenarios of the monthly aggregated NIE in the Northeastern region of Brazil illustrated in Fig. 4.1. This example employs a lognormal GAS model with tailored lag structure, identity scaling and time-varying parameter μ_t. Finally, NIE scenarios are simulated based on the fitted hyperparameters.

In this model we utilize lags 1 to 4 for the score and 1, 2, 3, 10, 11, 12 for the autoregressive components. Although the main goal of this experiment is to elucidate the software syntax, this model was chosen based on the Bayesian Information Criteria of candidate models. The resulting model can be written as follows:

$$\begin{cases} \mu_{t+1} &= \omega_1 + \sum_{i=1}^{4} A_i s_{t-i+1} + B_1\mu_t + B_2\mu_{t-1} + B_3\mu_{t-2} + \\ & \quad B_{10}\mu_{t-9} + B_{11}\mu_{t-10} + B_{12}\mu_{t-11}, \\ \ln(\sigma^2_{t+1}) &= \omega_2 \end{cases} \tag{4-1}$$

In this model the initial parameters $\mu_1, \ldots, \mu_{12}$, $\sigma^2_1, \ldots, \sigma^2_{12}$ are unknown and must be estimated. Although the parameters $\mu_1, \ldots, \mu_{12}$, $\sigma^2_1, \ldots, \sigma^2_{12}$ could be included in the vector of hyperparameters θ, this would add 24 extra

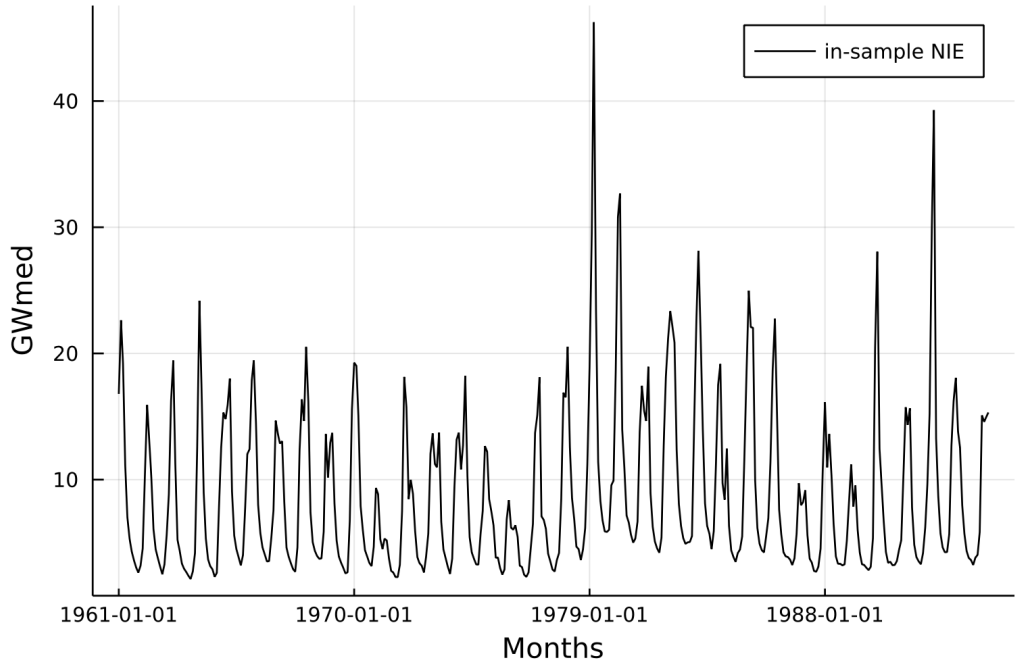

Figure 4.1: NIE in the Northeastern region of Brazil.

parameters to be estimated and thus significantly increase the computational cost of the estimation.

For this reason, different procedures to approximate the initial parameters are usually employed. In **ScoreDrivenModels.jl**, we implement the procedure described in [16]. The first step is to separate the original time series y_t into 12 time series containing solely the observations of same month, i.e., $\{y_1, y_{13}, y_{25}, \ldots, y_{1+12n}\}$, $\{y_2, y_{14}, y_{26}, \ldots, y_{2+12n}\}$, $\ldots$, $\{y_{12}, y_{24}, y_{36}, \ldots, y_{12+12n}\}$, and then the second step is to estimate the parameters $\mu_1, \ldots \mu_{12}$, $\sigma_1^2, \ldots, \sigma_{12}^2$ using maximum likelihood in the correspondent time series. This means that μ_2, σ_2^2 will be estimated using the time series $\{y_2, y_{14}, y_{26}, \ldots, y_{2+12n}\}$ and so on for other pairs of initial parameters. Note that this procedure, which is implemented by the function `dynamic_initial_params` in our package, is relevant for ensuring good estimation results when considering seasonal time series. The model is estimated as follows:

```
julia> Random.seed!(123)
julia> y = vec(readdlm("../test/data/nie_northeastern.csv"))
julia> y_train = y[1:400]
julia> gas = Model([1, 2, 3, 4], [1, 2, 3, 10, 11, 12],
                LogNormal, 1.0; time_varying_params=[1])
julia> initial_params = dynamic_initial_params(y_train, gas)
julia> f = ScoreDrivenModels.fit!(gas, y_train;
```

```
                                    initial_params=initial_params)
julia> plot(f)
```

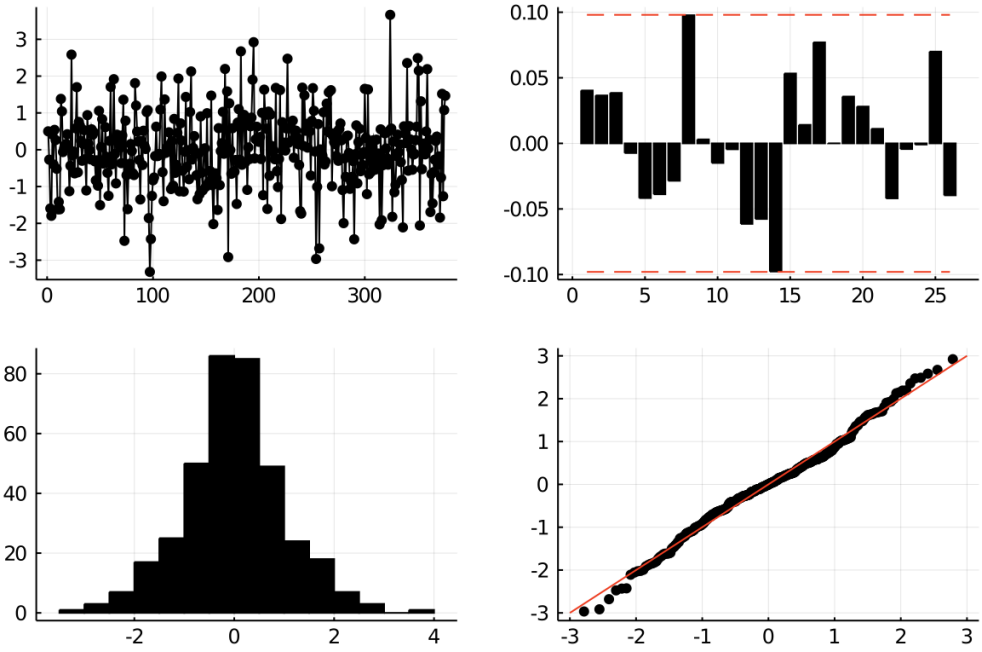

Figure 4.2: Quantile residuals of the log normal GAS with inverse scaling model. Top left - quantile residuals, top right - autocorrelation function, bottom left - histogram, bottom right - QQ plot.

Once the model is estimated we can generate 1000 NIE scenarios for the next 60 months following the methodology discussed in Section 2.3. We will utilize `forecast` in order to obtain the quantiles as well. The data set used to estimate the model used data from January 1961 until April 1994. To illustrate the adequacy of our model forecast, we present in Fig. 4.3 an out-of-sample study, where the simulated scenarios (gray lines), and related quantiles (red dotted lines) from May 1994 until April 1999 are contrasted to the actual observed values.

```
julia> forec = forecast(y_train, gas, 60;
                        S = 1_000, initial_params = initial_params)
```

4.2 GARCH model

One of the advanced features of **ScoreDrivenModels.jl** is allowing users to change the default parametrization. An example of a different parametrization is the GARCH(1, 1) GAS model. It can be shown that a GAS(1, 1) is equivalent to a GARCH(1, 1) if the function h is the identity (we refer the interested

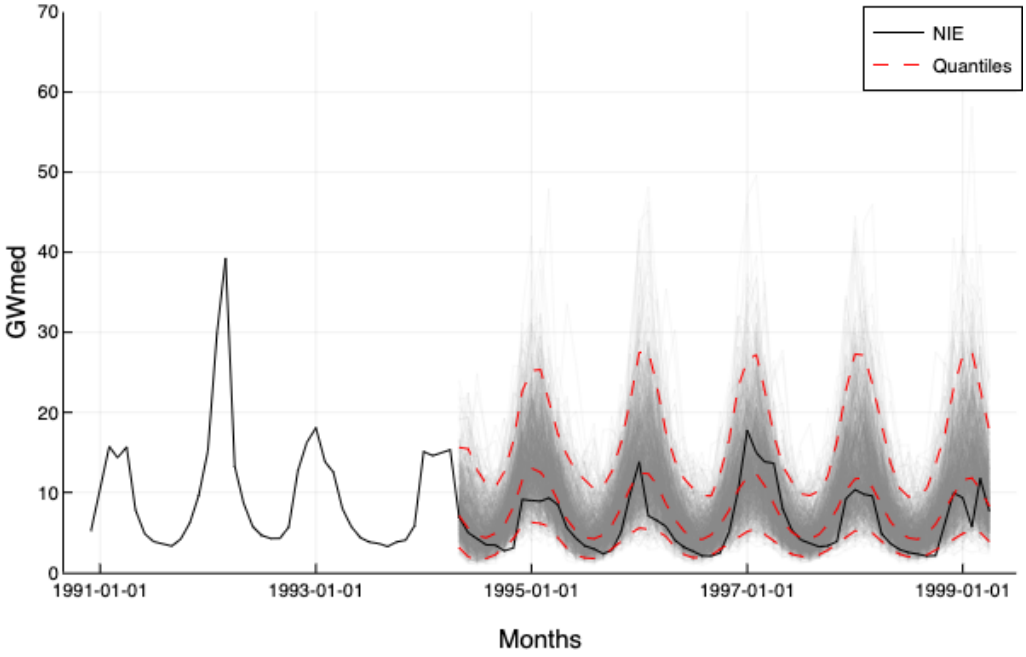

Figure 4.3: NIE scenarios in the Northeastern region of Brazil.

reader to Appendix A.3 for further details). To ensure the equivalence, the user must define the identity link function for both time-varying parameters. To do this the user must overwrite three **ScoreDrivenModel.jl** methods as detailed in the following example:

```
julia> function ScoreDrivenModels.link!(param_tilde::Matrix{T},
                                        ::Type{Normal}, param::Matrix{T},
                                        t::Int) where T
    param_tilde[t, 1] = link(IdentityLink, param[t, 1])
    param_tilde[t, 2] = link(IdentityLink, param[t, 2])
    return
end
julia> function ScoreDrivenModels.unlink!(param::Matrix{T},
                                          ::Type{Normal},
                                       param_tilde::Matrix{T},
                                        t::Int) where T
    param[t, 1] = unlink(IdentityLink, param_tilde[t, 1])
    param[t, 2] = unlink(IdentityLink, param_tilde[t, 2])
    return
end
julia> function ScoreDrivenModels.jacobian_link!(
                    aux::AuxiliaryLinAlg{T}, ::Type{Normal},
                    param::Matrix{T}, t::Int) where T
    aux.jac[1] = jacobian_link(IdentityLink, param[t, 1])
    aux.jac[2] = jacobian_link(IdentityLink, param[t, 2])
    return
end
```


Once the methods have been overwritten for the Normal distribution, the recursion will apply the `IdentityLink` in both parameters and the user can proceed to the estimation step. Below, we run an example provided in [29] for daily German mark/British pound exchange rates. Note that there are also bounds being provided for the hyperparameter estimation through `lb` and `ub`.

```
julia> y = vec(readdlm("../test/data/BG96.csv"))
julia> initial_params = [mean(y) var(y)]
julia> ub = [1.0, 1.0, 0.5, 1.0]
julia> lb = [-1.0, 0.0, 0.0, 0.5]
julia> gas = Model(1, 1, Normal, 1.0, time_varying_params = [2])
julia> initial_point = [0.0, 0.5, 0.25, 0.75]
julia> f = fit!(gas, y; initial_params = initial_params,
                        opt_method = IPNewton(gas, [initial_point]
                        ub=ub, lb=lb))
Round 1 of 1 - Log-likelihood: -1106.598367006442
julia> results(f)
```

```
---------------------------------------------------------
Distribution:                  Normal
Number of observations:        1974
Number of unknown parameters: 4
Log-likelihood:                -1106.5984
AIC:                           2221.1967
BIC:                           2243.5480
---------------------------------------------------------
Parameter      Estimate   Std.Error     t stat   p-value
omega_1         -0.0062      0.0085    -0.7373    0.5019
omega_2          0.0108      0.0029     3.7732    0.0196
A_1_22           0.1534      0.0266     5.7726    0.0045
B_1_22           0.9593      0.0144    66.6015    0.0000
```

The results obtained above are the same as the ones found in the Section 4.1 of [29], with the exception of one parameter. This is due to a difference in the parametrizations of the models – the demonstration of the equivalence between the hyperparameters of the GAS model and the GARCH model can be found in Appendix A.3. Note that, following the demonstration in A.3, we have $\beta_1 = B_1 - A_1 = 0.8059$, which is exactly the reported value in [29].

We have also run the same example using the R package **rugarch** [30] and obtained the same result, thus further illustrating the equivalence of the GARCH(1, 1) and the Normal GAS(1, 1) under this parametrization.

5
Conclusion and discussion

The **ScoreDrivenModels.jl** package provides a general framework for score-driven models and represents a user-friendly, off-the-shelf tool. The package is fully implemented in Julia, thus not depending on subroutines written in lower-level languages such as C or Fortran. The model specification is flexible and allows defining any desired lag structure, as well as any distribution due to a tailored dependency on the **Distributions.jl** package. The estimation procedure is based on numerical optimization algorithms such as Nelder-Mead or L-BFGS and employs the well-known package **Optim.jl**. Special initialization procedures are implemented to robustify the estimation process for the case of seasonal time series. Available forecasting and simulation procedures allow users to study future data from the estimated model. Finally, the examples provided in Section 4 illustrate the functionalities of the package as well as possible applications. In particular, we discussed a time series model that only generates positive scenarios, which is a very desirable characteristic for this application and a source of many discussions throughout the Brazilian energy sector. The software continues to evolve, new features such as an heuristic for the initial hyperparameter and unobserved components modeling are considered as future research topics. The software documentation can be found in `https://lampspuc.github.io/ScoreDrivenModels.jl/latest/`

During the conception of this work, we came across the many challenges of implementing score-driven models. We would like to list insights gained from experience as well as challenges that are yet to be faced, also known as research opportunities

- Insights
  - We found that the choice of initial parameters of the updating mechanism significantly influences the estimation procedure and that efficient initial parameters choices are problem-dependent. In particular, it is a good practice for seasonal time series to use heuristics based on seasonal unconditional statistics such as the one described in Section 4.
  - We understand that some models diverge without a proper scaling coefficient d. Even when choosing the correct coefficient d, the

Fisher information might not be the best scaling matrix. Also, it is not easy to calculate the Fisher information analytically for some distributions, and one example is the Weibull distribution. An alternative to the Fisher information is to scale using the Hessian matrix or a moving average of Hessian matrices.
  - Empirically we observe that slightly different initial points on the log-likelihood maximization can lead to very different objective values. We tackled this problem by running the maximization with different initial points and analyzing if they converge to the same objective.

- Future research topics

  - Study how the consideration of initial parameters as part of the hyperparameters vector affect the estimation procedure. Some interesting questions: Considering the initial parameters as hyperparameters improve the maximum likelihood considerably? Considering the initial parameters as hyperparameters improve the forecasting performance of score-driven models? In other words, does it cause overfitting? When considering initial parameters as hyperparameters, what should be the heuristics to choose initial optimization points?
  - Compare the forecasting performance of different updating mechanisms, e.g., GAS(p, q) dynamics against unobserved components dynamics.
  - Extensively study and compare heuristics for choosing the initial points in the optimization routine. Implement estimation procedures that use automatic differentiation techniques and analyze if there are any differences in speed and accuracy compared to estimation procedures that use finite difference techniques.

A
Notes on parametrizations

The use of different link functions and scaling coefficients gives rise to different parametrizations and, therefore, different models. In this appendix, we present and explore modeling variants due to the choice of link functions and values for the scaling coefficient.

A.1
Possible Parametrizations

There are three mapping functions available in **ScoreDrivenModels.jl**. Within the context of the software we call them `Links`. Each one of them can be used as the default mapping function for a given distribution implemented in the package.

- Identity link: $\tilde{f} = f$ where $f \in \mathbb{R}$ and $\tilde{f} \in \mathbb{R}$
- Log link: $\tilde{f} = \ln(f - a)$ where $f \in [a, \infty)$ and $\tilde{f} \in \mathbb{R}$
- Logit link: $\tilde{f} = \ln\left(\frac{f-a}{b-f}\right)$ where $f \in [a, b]$ and $\tilde{f} \in \mathbb{R}$

A.2
Score Derivations for Different Scaling Values

In this section we derive expressions (2-9), (2-10), and (2-11).

A.2.1
Scaling d = 0

For $d = 0$ the score is simply equal to

$$\tilde{\nabla}_t = \frac{\partial \ln p(y_t|y_{t-1}, f_t)}{\partial \tilde{f}_t}, \tag{A-1}$$

which is equivalent to

$$\tilde{\nabla}_t = \frac{\partial f_t}{\partial \tilde{f}_t} \cdot \frac{\partial \ln p(y_t|y_{t-1}, f_t)}{\partial f_t}. \tag{A-2}$$

Notice that one can show by the inverse function theorem that this is the inverse of the Jacobian $\dot{h}$. Thus, it follows that

$$\tilde{\nabla}_t = \left(\dot{h}\right)^{-1} \nabla_t. \tag{A-3}$$

By proceeding the calculus above indicated, we show that

$$\tilde{\nabla}_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \tilde{\sigma}^2} \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sigma_t^2} \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \end{bmatrix} \tag{A-4}$$

$$\tilde{\nabla}_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \tilde{\mu}} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \tilde{\sigma}^2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \sigma_t^2 \end{bmatrix} \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \end{bmatrix} \tag{A-5}$$

$$\tilde{\nabla}_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \tilde{\mu}} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \tilde{\sigma}^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \cdot \sigma_t^2 \end{bmatrix}. \tag{A-6}$$

A.2.2
Scaling d = 1/2

The original scaled score is $s_t = \mathcal{J}_{t|t-1}\nabla_t$ as in considered in [5]. Thus, recall that

$$\mathcal{I}_{t|t-1}^{-1} = \mathcal{J}_{t|t-1}\mathcal{J}_{t|t-1}^{\top}. \tag{A-7}$$

In this case, the new scaled score is $\tilde{s}_t = \tilde{\mathcal{J}}_{t|t-1}\tilde{\nabla}_t$. Thus, if $\tilde{\nabla}_t = \left(\dot{h}\right)^{-1}\nabla_t$ and $\tilde{\mathcal{J}}_{t|t-1}$ is yet to be calculated, we have

$$\tilde{\mathcal{I}}_{t|t-1} = E\left[\nabla_t \nabla_t^{\top}\right] \tag{A-8}$$

$$\tilde{\mathcal{I}}_{t|t-1} = E\left[\left(\dot{h}\right)^{-1}\nabla_t \nabla_t^{\top}\left(\dot{h}\right)^{-1}\right] \tag{A-9}$$

$$\tilde{\mathcal{I}}_{t|t-1} = \left(\dot{h}\right)^{-1} E\left[\nabla_t \nabla_t^{\top}\right]\left(\dot{h}\right)^{-1} \tag{A-10}$$

$$\tilde{\mathcal{I}}_{t|t-1} = \left(\dot{h}\right)^{-1}\mathcal{I}_{t|t-1}\left(\dot{h}\right)^{-1}. \tag{A-11}$$

As $\left(\dot{h}\right)^{-1}$ is a diagonal matrix, in the above development we omitted the transpose operator. So, the inverse of the reparametrized information matrix is equal to

$$\tilde{\mathcal{I}}_{t|t-1}^{-1} = \dot{h}\mathcal{I}_{t|t-1}^{-1}\dot{h} \tag{A-12}$$

$$\tilde{\mathcal{I}}_{t|t-1}^{-1} = \dot{h}\mathcal{J}_{t|t-1}\mathcal{J}_{t|t-1}^{\top}\dot{h}. \tag{A-13}$$

Hence, it follows that

$$\tilde{\mathcal{J}}_{t|t-1}\tilde{\mathcal{J}}_{t|t-1}^{\top} = \dot{h}\mathcal{J}_{t|t-1}\mathcal{J}_{t|t-1}^{\top}\dot{h} \tag{A-14}$$

$$\tilde{\mathcal{J}}_{t|t-1} = \dot{h}\mathcal{J}_{t|t-1}, \tag{A-15}$$

which lead us to conclude that for this type of scaling $\tilde{s}_t = s_t$. This becomes clear in the following development:

$$\tilde{s}_t = \tilde{\mathcal{J}}_{t|t-1}\tilde{\nabla}_t \tag{A-16}$$

$$\tilde{s}_t = \dot{h}\mathcal{J}_{t|t-1}\left(\dot{h}\right)^{-1}\nabla_t, \tag{A-17}$$

In the case that $\mathcal{J}_{t|t-1}$ is diagonal, because $\dot{h}$ is diagonal we have

$$\tilde{s}_t = \mathcal{J}_{t|t-1}\nabla_t = s_t. \tag{A-18}$$

A.2.3
Scaling d = 1

In this case, the scaled score is equal to $s_t = \mathcal{I}^{-1}_{t|t-1}\nabla_t$ and the reparametrized scaled score is equal to $\tilde{s}_t = \tilde{\mathcal{I}}^{-1}_{t|t-1}\tilde{\nabla}_t$. As previously calculated we have that

$$\tilde{\mathcal{I}}^{-1}_{t|t-1} = \dot{h}\mathcal{I}^{-1}_{t|t-1}\dot{h} \tag{A-19}$$

$$\tilde{\nabla}_t = \left(\dot{h}\right)^{-1}\nabla_t. \tag{A-20}$$

Therefore,

$$\tilde{s}_t = \tilde{\mathcal{I}}^{-1}_{t|t-1}\tilde{\nabla}_t \tag{A-21}$$

$$\tilde{s}_t = \dot{h}\mathcal{I}^{-1}_{t|t-1}\dot{h}\left(\dot{h}\right)^{-1}\nabla_t \tag{A-22}$$

$$\tilde{s}_t = \dot{h}\mathcal{I}^{-1}_{t|t-1}\nabla_t \tag{A-23}$$

$$\tilde{s}_t = \dot{h}s_t. \tag{A-24}$$

A.3
Different Parametrizations Lead to Different Models

One of the examples given in [5] shows the equivalence between a GARCH(1, 1) and GAS(1, 1) with Normal distribution and $d = 1$. An important note on this fact is that the models are only equivalent if the variance, σ^2, is considered a time-varying parameter, i.e., if the link is the identity function. Therefore, if a log link is used to ensure a positive value for σ^2, for instance, the equivalence will not hold. As this calculations are not shown in any other work and they reveal relevant insights that can be tested in through our software, we provide further details in the sequel.

Thus, let us develop the GAS recursion for both cases. Recall the Normal probability density function (pdf):

$$p(y_t|y_{t-1},\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{\left(\frac{-(y_t-\mu)^2}{2\sigma^2}\right)}. \tag{A-25}$$

The log-pdf is:

$$\ln p(y_t|y_{t-1},\mu,\sigma^2) = \frac{-1}{2}\ln 2\pi - \frac{1}{2}\ln\sigma^2 - \frac{1}{2}\frac{(y_t-\mu)^2}{\sigma^2}. \tag{A-26}$$

To calculate the score with respect to σ^2 we need to calculate the following derivative:

$$\frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial\sigma^2} = \frac{-1}{2\sigma^2} + \frac{(y_t-\mu)^2}{2\sigma^4}. \tag{A-27}$$

Then, we calculate the Fisher information as follows:

$$-E\left[\frac{\partial^2 \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial\sigma^2\partial\sigma^2}\right] = \frac{1}{2\sigma^4}. \tag{A-28}$$

Now if we proceed to write the GAS(1,1) recursion using the inverse scaling $d = 1$, we find

$$\sigma^2_{t+1} = \omega + A_1 s_t + B_1\sigma^2_t \tag{A-29}$$

$$\sigma^2_{t+1} = \omega + A_1((y_t-\mu_t)^2 - \sigma^2_t) + B_1\sigma^2_t. \tag{A-30}$$

By assuming $\mu = 0$ under correct specification, the recursion becomes

$$\sigma^2_{t+1} = \omega + A_1(y^2_t - \sigma^2_t) + B_1\sigma^2_t, \tag{A-31}$$

which is equivalent to the GARCH(1,1) model

$$\sigma^2_{t+1} = \alpha_0 + \alpha_1 y^2_t + \beta_1\sigma^2_t. \tag{A-32}$$

Note that there is sufficient degrees of freedom to make the correspondence between the parameters of the two models, e.g., $\alpha_0 = \omega, \alpha_1 = A_1, \beta_1 = B_1 - A_1$.

Now let us work with a different parametrization to assure a positive value to σ^2. The approach suggested in [5] is to use a map $h(\cdot) = \ln(\cdot)$, i.e., $\tilde{\sigma}^2 = \ln\sigma^2$. When we use this parametrization the recursion adapts the following way:

$$\begin{cases} \sigma^2_t &= h^{-1}(\tilde{\sigma}^2_t), \\ \tilde{\sigma}^2_{t+1} &= \omega + A_1\tilde{s}_t + B_1\tilde{\sigma}^2_t. \end{cases} \tag{A-33}$$

And we shown in the previous section that $\tilde{s}_t = \dot{h}s_t$. In this case $\dot{h} = \frac{1}{\sigma^2_t}$, so the recursion assumes the following form:

$$\begin{cases} \sigma^2_t &= h^{-1}(\tilde{\sigma}^2_t), \\ \tilde{\sigma}^2_{t+1} &= \omega + A_1\frac{y^2_t-\sigma^2_t}{\sigma^2_t} + B_1\tilde{\sigma}^2_t. \end{cases} \tag{A-34}$$

If we rewrite the recursion solely in terms of σ^2 we have

$$\ln\left(\sigma^2_{t+1}\right) = \omega + A_1\frac{y^2_t-\sigma^2_t}{\sigma^2_t} + B_1\ln\left(\sigma^2_t\right). \tag{A-35}$$

In this case however, it is impossible to choose the parameters ω, A_1 and B_1 to meet a recursion equivalent to (A-32).

A.4
Imprecision in the R GAS package

In [18], they develop the relations for the score and Fisher information when dealing with reparametrizations. The relations that they state are correct but incomplete for the understanding of the GAS updating mechanism. In fact, these missing informations are mistakenly programmed in their package. This mistakes cause problems in cases where users request for calls where the mapping is not the identity mapping and the scaling is either the `"Inv"` or `"InvSqrt"`. To be more specific, let us explain the error with the inverse scaling. In their work they state that:

$$\tilde{\mathcal{I}}_{t|t-1} = \left(\dot{h}\right)^{-1} \mathcal{I}_{t|t-1} \left(\dot{h}\right)^{-1}. \tag{A-36}$$

which is correct, but they do not mention what should be the expression for $\tilde{\mathcal{I}}^{-1}_{t|t-1}$. The correct expression for $\tilde{\mathcal{I}}^{-1}_{t|t-1}$ is:

$$\tilde{\mathcal{I}}^{-1}_{t|t-1} = \left(\dot{h}\right) \mathcal{I}^{-1}_{t|t-1} \left(\dot{h}\right). \tag{A-37}$$

but the programmed on in their work is currently:

$$\tilde{\mathcal{I}}^{-1}_{t|t-1} = \left(\dot{h}\right)^{-1} \mathcal{I}^{-1}_{t|t-1} \left(\dot{h}\right)^{-1}. \tag{A-38}$$

leading to a wrong $\tilde{s}_t$. The correct expression for s_t under inverse scaling should be $\dot{h}\tilde{\mathcal{I}}^{-1}_{t|t-1}\nabla_t$ but the programmed one on [18] is $\left(\dot{h}\right)^{-1} \mathcal{I}^{-1}_{t|t-1} \left(\dot{h}\right)^{-1} \left(\dot{h}\right)^{-1} \nabla_t$. A very similar mistake occurs for the inverse square root scaling, this time the correct one should be $\mathcal{J}_{t|t-1}\nabla_t$ but the currently programmed one is $\mathcal{J}_{t|t-1} \left(\dot{h}\right)^{-1} \nabla_t$.

B
Scores

GAS models can be still considered a relatively recent technology. The derivation of the analytical form of the score is hard to find in the literature or simply nonexistent. In this context, this paper also aims to provide users with a technical reference for the presented software. Therefore, in this Appendix, we provide detailed information on the score calculation for each distribution considered in the package.

B.1
Beta

Density function

$$p(y_t|y_{t-1},\alpha,\beta) = \frac{y_t^{\alpha-1}(1-y_t)^{\beta-1}}{B(\alpha,\beta)} \quad \text{where} \quad B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)},$$
$$\alpha \in \mathbb{R}^+, \quad \beta \in \mathbb{R}^+, \quad \Gamma(\cdot) \text{ is the Gamma function.}$$

$$\mathsf{E}[y_t|y_{t-1}] = \frac{\alpha}{\alpha+\beta},$$

$$\mathsf{VAR}[y_t|y_{t-1}] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\alpha,\beta)}{\partial\alpha} \\ \frac{\partial \ln p(y_t|y_{t-1},\alpha,\beta)}{\partial\beta} \end{bmatrix}$$

$$\ln p(y_t|y_{t-1},a,c,\alpha,\beta) = (\alpha-1)\ln y_t + (\beta-1)\ln(1-y_t) - \ln B(\alpha,\beta)$$

$$\nabla_t^\alpha = \frac{\partial \ln p(y_t|y_{t-1},\alpha,\beta)}{\partial\alpha} = \ln y_t + \psi(\alpha+\beta) - \psi(\alpha)$$

$$\nabla_t^\beta = \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial\beta} = \ln(1-y_t) + \psi(\alpha+\beta) - \psi(\beta)$$

B.2

Beta four parameters

Density function

$$p(y_t|y_{t-1},a,c,\alpha,\beta) = \frac{(y_t-a)^{\alpha-1}\,(c-y_t)^{\beta-1}}{(c-a)\,B(\alpha,\beta)} \quad \text{where} \quad B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)},$$
$$a,c \in \mathbb{R} \quad \alpha,\beta \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = a + (c-a)\,\frac{\alpha}{\alpha+\beta},$$
$$\mathsf{VAR}[y_t|y_{t-1}] = (c-a)^2\,\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial a} \\ \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial c} \\ \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial \alpha} \\ \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial \beta} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1},a,c,\alpha,\beta) = (\alpha-1)\ln(y_t-a) + (\beta-1)\ln(c-y_t) - (\alpha+\beta-1)\ln(c-a) - \ln B(\alpha,\beta)$$
$$\nabla_t^a = \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial a} = \frac{-\alpha+1}{y-a} + \frac{\alpha+\beta-1}{c-a}$$
$$\nabla_t^c = \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial c} = \frac{\beta-1}{c-y} - \frac{\alpha+\beta-1}{c-a}$$
$$\nabla_t^\alpha = \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial \alpha} = \ln(y_t-a) - \ln(c-a) + \psi(\alpha+\beta) - \psi(\alpha)$$
$$\nabla_t^\beta = \frac{\partial \ln p(y_t|y_{t-1},a,c,\alpha,\beta)}{\partial \beta} = \ln(c-y_t) - \ln(c-a) + \psi(\alpha+\beta) - \psi(\beta)$$

B.3 Exponential

Density function

$$p(y_t|y_{t-1},\lambda) = \lambda e^{-\lambda y_t} \quad \lambda \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = \frac{1}{\lambda},$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \frac{1}{\lambda^2}$$

Score calculation

$$\nabla_t = \left[\frac{\partial \ln p(y_t|y_{t-1},\lambda)}{\partial \lambda}\right]$$
$$\ln p(y_t|y_{t-1},\lambda,k) = \ln\lambda - e^{-\lambda y_t}$$

$$\nabla_t^{\lambda} = \frac{\partial \ln p(y_t|y_{t-1},\lambda)}{\partial \lambda} = \frac{1}{\lambda} - y_t$$

B.4 Gamma

Density function

$$p(y_t|y_{t-1},\alpha,k) = \frac{y_t^{\alpha-1}e^{-\frac{y_t}{k}}}{\Gamma(\alpha)k^{\alpha}}, \quad \alpha \in \mathbb{R}^+, \quad k \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = \alpha k,$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \alpha k^2$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\alpha,k)}{\partial \alpha} \\ \frac{\partial \ln p(y_t|y_{t-1},\alpha,k)}{\partial k} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1},\alpha,k) = (\alpha-1)\ln y_t - \frac{y_t}{k} - \ln\Gamma(\alpha) - \alpha \ln k$$
$$\nabla_t^{\alpha} = \frac{\partial \ln p(y_t|y_{t-1},\alpha,k)}{\partial \alpha} = \ln y_t - \psi(\alpha) - \ln k$$
$$\nabla_t^{k} = \frac{\partial \ln p(y_t|y_{t-1},\alpha,k)}{\partial k} = \frac{y_t}{k^2} - \frac{\alpha}{k}$$

B.5 Logit-Normal

Density function

$$p(y_t|y_{t-1},\mu,\sigma^2) = \frac{1}{y_t(1-y_t)\sqrt{2\pi\sigma^2}}\, e^{\left(-\frac{(\text{logit}(y_t)-\mu)^2}{2\sigma^2}\right)}, \quad \mu \in \mathbb{R}, \quad \sigma^2 \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = e^{\left(\mu+\frac{\sigma^2}{2}\right)},$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \left(e^{\sigma^2}-1\right)e^{\left(2\mu+\sigma^2\right)}$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1},\mu,\sigma^2) = -\ln\left(y_t(1-y_t)\right) - \frac{1}{2}\ln 2\pi\sigma^2 - \frac{(\text{logit}\,(y_t)-\mu)^2}{2\sigma^2}$$
$$\nabla_t^{\mu} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} = \frac{\text{logit}\,(y_t)-\mu}{\sigma^2}$$

$$\nabla_t^{\sigma^2} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} = \frac{-1}{2\sigma^2}\left(1 - \frac{\text{logit}\,(y_t) - \mu)^2}{\sigma^2}\right)$$

B.6 Lognormal

Density function

$$p(y_t|y_{t-1},\mu,\sigma^2) = \frac{1}{y_t\sqrt{2\pi\sigma^2}}\, e^{\left(-\frac{(\ln y_t-\mu)^2}{2\sigma^2}\right)}, \quad \mu \in \mathbb{R}, \quad \sigma^2 \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = e^{\left(\mu+\frac{\sigma^2}{2}\right)},$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \left(e^{\sigma^2}-1\right)e^{\left(2\mu+\sigma^2\right)}$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1},\mu,\sigma^2) = -\ln y_t - \frac{1}{2}\ln 2\pi\sigma^2 - \frac{(\ln y_t-\mu)^2}{2\sigma^2}$$
$$\nabla_t^{\mu} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} = \frac{\ln y_t - \mu}{\sigma^2}$$
$$\nabla_t^{\sigma^2} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} = \frac{-1}{2\sigma^2}\left(1 - \frac{(\ln y_t-\mu)^2}{\sigma^2}\right)$$

B.7 Negative binomial

Density function

$$p(y_t|y_{t-1},r,p) = \frac{\Gamma(y_t+r)}{y_t!\Gamma(r)}p^r\,(1-p)^{y_t}, \quad r \in \mathbb{R}^+, \quad p \in [0,1]$$
$$\mathsf{E}[y_t|y_{t-1}] = \frac{pr}{1-p},$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \frac{pr}{(1-p)^2}$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},r,p)}{\partial r} \\ \frac{\partial \ln p(y_t|y_{t-1},r,p)}{\partial p} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1},r,p) = \ln\Gamma(y_t+r) - (\ln y_t!\Gamma(r)) + r\ln p + y_t\ln(1-p)$$
$$\nabla_t^{r} = \frac{\partial \ln p(y_t|y_{t-1},r,p)}{\partial r} = \psi(y_t+r) - \psi(r) + \ln p$$

$$\nabla_t^p = \frac{\partial \ln p(y_t|y_{t-1}, r, p)}{\partial p} = \frac{r}{p} - \frac{k}{1-p}$$

B.8 Normal

Density function

$$p(y_t|y_{t-1}, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(y_t-\mu)^2}{2\sigma^2}\right)}, \quad \mu \in \mathbb{R}, \quad \sigma^2 \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = \mu,$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \sigma^2$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2)}{\partial \sigma^2} \end{bmatrix}$$
$$\ln p(y_t|y_{t-1}, \mu, \sigma^2) = \frac{-1}{2} \ln 2\pi\sigma^2 - \frac{(y_t-\mu)^2}{2\sigma^2}$$
$$\nabla_t^\mu = \frac{\partial \ln p(y_t|y_{t-1}, \mu, \sigma^2)}{\partial \mu} = \frac{y_t - \mu}{\sigma^2}$$
$$\nabla_t^{\sigma^2} = \frac{\partial \ln p(y_t|y_{t-1}, \mu, \sigma^2)}{\partial \sigma^2} = \frac{-1}{2\sigma^2}\left(1 - \frac{(y_t-\mu)^2}{\sigma^2}\right)$$

B.9 Poisson

Density function

$$p(y_t|y_{t-1}, \lambda) = \frac{e^{-\lambda}\lambda^{y_t}}{y_t!}, \quad \lambda \in \mathbb{R}^+$$
$$\mathsf{E}[y_t|y_{t-1}] = \lambda,$$
$$\mathsf{VAR}[y_t|y_{t-1}] = \lambda$$

Score calculation

$$\nabla_t = \frac{\partial \ln p(y_t|y_{t-1}, \lambda)}{\partial \lambda}$$
$$\ln p(y_t|y_{t-1}, \lambda) = -\lambda + y_t \ln \lambda - \ln y_t!$$
$$\nabla_t^\lambda = \frac{\partial \ln p(y_t|y_{t-1}, \lambda)}{\partial \lambda} = \frac{y_t - \lambda}{\lambda}$$

B.10
Student's t

Density function

$$p(y_t|y_{t-1},\nu) = \frac{1}{\sqrt{\nu}B\left(\frac{1}{2},\frac{\nu}{2}\right)}\left(1+\frac{y_t^2}{\nu}\right)^{\frac{-\nu+1}{2}}, \quad \nu \in \mathbb{R}^+$$

$\mathsf{E}[y_t|y_{t-1}] = 0,$

$\mathsf{VAR}[y_t|y_{t-1}] = \sigma^2\frac{\nu}{\nu-2}$ for $\nu > 2$, ∞ for $1 < \nu \leq 2$, undefined otherwise.

Score calculation

$$\nabla_t = \left[\frac{\partial \ln p(y_t|y_{t-1},\nu)}{\partial \nu}\right]$$

$$\ln p(y_t|y_{t-1},\nu) = -\frac{1}{2}\ln\nu - \ln B\left(\frac{1}{2},\frac{\nu}{2}\right) - \left(\frac{\nu+1}{2}\right)\ln\left(1+\frac{y_t^2}{\nu}\right)$$

$$\nabla_t^{\nu} = \frac{\partial \ln p(y_t|y_{t-1}\nu)}{\partial \nu} = \frac{1}{2}\left(\frac{(\nu+1)\,y_t^2}{\nu y_t^2+\nu^2} - \frac{1}{\nu} - \ln\left(\frac{y_t^2}{\nu}+1\right) + \psi\left(\frac{\nu+1}{2}\right) - \psi\left(\frac{\nu}{2}\right)\right)$$

B.11
Student's t with Location and Scale

Density function

$$p(y_t|y_{t-1},\mu,\sigma^2,\nu) = \frac{1}{\sqrt{\sigma^2\nu}B\left(\frac{1}{2},\frac{\nu}{2}\right)}\left(1+\frac{(y_t-\mu)^2}{\sigma^2\nu}\right)^{\frac{-\nu+1}{2}}, \quad \mu \in \mathbb{R},$$

$$\sigma^2 \in \mathbb{R}^+ \quad \nu \in \mathbb{R}^+$$

$\mathsf{E}[y_t|y_{t-1}] = \mu,$

$\mathsf{VAR}[y_t|y_{t-1}] = \sigma^2\frac{\nu}{\nu-2}$ for $\nu > 2$, ∞ for $1 < \nu \leq 2$, undefined otherwise.

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \mu} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \sigma^2} \\ \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \nu} \end{bmatrix}$$

$$\ln p(y_t|y_{t-1},\mu,\sigma^2,\nu) = -\frac{1}{2}\ln\nu\sigma^2 - \ln B\left(\frac{1}{2},\frac{\nu}{2}\right) - \left(\frac{\nu+1}{2}\right)\ln\left(1+\frac{(y_t-\mu)^2}{\sigma^2\nu}\right)$$

$$\nabla_t^{\mu} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \mu} = \frac{(\nu+1)\,(y_t-\mu)}{(y_t-\mu)^2+\sigma^2\nu}$$

$$\nabla_t^{\sigma^2} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \sigma^2} = -\nu \frac{\sigma^2 - (y_t-\mu)^2}{2\sigma^2\left(\nu\sigma^2 + (y_t-\mu)^2\right)}$$

$$\nabla_t^{\nu} = \frac{\partial \ln p(y_t|y_{t-1},\mu,\sigma^2,\nu)}{\partial \nu} = \frac{1}{2}\left(\frac{(\nu+1)(y_t-\mu)^2}{\nu(y_t-\mu)^2 + \sigma^2\nu^2} - \frac{1}{\nu} - \ln\left(\frac{(y_t-\mu)^2}{\sigma^2\nu}+1\right)\right) + \frac{1}{2}\left(\psi\left(\frac{\nu+1}{2}\right) - \psi\left(\frac{\nu}{2}\right)\right)$$

B.12 Weibull

Density function

$$p(y_t|y_{t-1},\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{y_t}{\lambda}\right)^{k-1} e^{\left(-\frac{y_t}{\lambda}\right)^k} & x \geq 0, \\ 0 & x < 0, \end{cases} \quad \lambda \in \mathbb{R}^+, k \in \mathbb{R}^+$$

$$\mathsf{E}[y_t|y_{t-1}] = \lambda\Gamma\left(1+1/k\right),$$

$$\mathsf{VAR}[y_t|y_{t-1}] = \lambda^2\left[\Gamma\left(1+\frac{2}{k}\right) - \left(\Gamma\left(1+\frac{1}{k}\right)\right)^2\right]$$

Score calculation

$$\nabla_t = \begin{bmatrix} \frac{\partial \ln p(y_t|y_{t-1},\lambda,k)}{\partial k} \\ \frac{\partial \ln p(y_t|y_{t-1},\lambda,k)}{\partial \lambda} \end{bmatrix}$$

$$\ln p(y_t|y_{t-1},\lambda,k) = \ln k + (k-1)\ln y_t - k\ln\lambda - \left(\frac{y_t}{\lambda}\right)^k$$

$$\nabla_t^{\lambda} = \frac{\partial \ln p(y_t|y_{t-1},\lambda,k)}{\partial \lambda} = \frac{k}{\lambda}\left(\left(\frac{y_t}{\lambda}\right)^k - 1\right)$$

$$\nabla_t^{k} = \frac{\partial \ln p(y_t|y_{t-1},\lambda,k)}{\partial k} = \frac{1}{k} + \ln\left(\frac{y_t}{\lambda}\right)\left(1 - \left(\frac{y_t}{\lambda}\right)^k\right)$$

C
Computational Details

The results in this paper were obtained using Julia 1.4.2 and **ScoreDrivenModels** v0.1.10. Figures were generated with **Plots.jl** v1.4.3. In order to guarantee that the results of the examples are exactly the same as the ones reported, we recommend the interested user to `activate` the `Project.toml` in the repository's example folder using Julia's **Pkg** manager.

D
References

[1] D. R. Cox, G. Gudmundsson, G. Lindgren, L. Bondesson, E. Harsaae, P. Laake, K. Juselius, and S. L. Lauritzen, "Statistical analysis of time series: Some recent developments [with discussion and reply]," *Scandinavian Journal of Statistics*, vol. 8, no. 2, pp. 93–115, 1981. [Online]. Available: http://www.jstor.org/stable/4615819

[2] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*. Oxford University Press, 2012.

[3] S. J. Koopman, A. C. Harvey, J. A. Doornik, and N. Shephard, "STAMP 6.0: Structural time series analyser, modeller and predictor," *London: Timberlake Consultants*, 2000.

[4] R. Saavedra, G. Bodin, and M. Souto, "Statespacemodels.jl: a julia package for time-series analysis in a state-space framework," *arXiv preprint arXiv:1908.01757*, 2019.

[5] D. Creal, S. J. Koopman, and A. Lucas, "Generalized autoregressive score models with applications," *Journal of Applied Econometrics*, vol. 28, no. 5, pp. 777–795, aug 2013.

[6] A. C. Harvey, *Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series*, ser. Econometric Society Monographs. Cambridge University Press, 2013.

[7] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[8] R. F. Engle and J. R. Russell, "Autoregressive conditional duration: A new model for irregularly spaced transaction data," *Econometrica*, vol. 66, no. 5, pp. 1127–1162, 1998. [Online]. Available: http://www.jstor.org/stable/2999632

[9] A. C. Harvey and S. Thiele, "Testing against changing correlation," *Journal of Empirical Finance*, vol. 38, no. Part B, pp. 575–589, sep 2016.

[10] A. Ayala and S. Blazsek, "Score-driven copula models for portfolios of two risky assets," *The European Journal of Finance*, vol. 24, no. 18, pp. 1861–1884, 2018.

[11] C. Neves, C. Fernandes, and H. Hoeltgebaum, "Five different distributions for the Lee-Carter model of mortality forecasting: A comparison using GAS models," *Insurance: Mathematics and Economics*, vol. 75, pp. 48–57, Jul. 2017.

[12] M. A. B. de Melo, C. A. Fernandes, and E. F. de Melo, "Forecasting aggregate claims using score-driven time series models," *Statistica Neerlandica*, vol. 72, no. 3, pp. 354–374, 2018.

[13] A. J. Patton, J. F. Ziegel, and R. Chen, "Dynamic semiparametric models for expected shortfall (and value-at-risk)," *Journal of Econometrics*, vol. 211, no. 2, pp. 388–413, 2019.

[14] A. Nani, I. Gamoudi, and M. El Ghourabi, "Value-at-risk estimation by ls-svr and fs-ls-svr based on gas model," *Journal of Applied Statistics*, vol. 46, no. 12, pp. 2237–2253, 2019.

[15] R. Saavedra, "A study on the impact of El Niño Southern Oscillation on hydro power generation in Brazil," 2017.

[16] H. Hoeltgebaum, C. Fernandes, and A. Street, "Generating joint scenarios for renewable generation: The case for non-gaussian models with time-varying parameters," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7011–7019, nov 2018.

[17] R. Taylor, "PyFlux," [Online]. Available: https://github.com/RJT1990/pyflux, 2016.

[18] D. Ardia, K. Boudt, and L. Catania, "Generalized autoregressive score models in R: The GAS package," *Journal of Statistical Software*, vol. 88, no. 1, 2019.

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[20] G. Bodin, R. Saavedra, and C. Fernandes, "`ScoreDrivenModels.jl` [Online]. Available: https://github.com/LAMPSPUC/ScoreDrivenModels.jl," 2020.

[21] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[22] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 01 1965. [Online]. Available: https://doi.org/10.1093/comjnl/7.4.308

[23] F. Blasques, S. J. Koopman, K. Łasak, and A. Lucas, "In-sample confidence bands and out-of-sample forecast bands for time-varying parameters in observation-driven models," *International Journal of Forecasting*, vol. 32, no. 3, pp. 875–887, Jul. 2016.

[24] L. Kalliovirta, "Misspecification tests based on quantile residuals," *The Econometrics Journal*, vol. 15, no. 2, pp. 358–393, 2012.

[25] S. Kolassa, "Evaluating predictive count data distributions in retail sales forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 788–803, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207016000315

[26] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management Science*, vol. 22, no. 10, pp. 1087–1096, 1976. [Online]. Available: https://EconPapers.repec.org/RePEc:inm:ormnsc:v:22:y:1976:i:10:p:1087-1096

[27] A. Jordan, F. Kruger, and S. Lerch, "Evaluating probabilistic forecasts with scoringrules." *arXiv: Computation*, 2017.

[28] P. K. Mogensen and A. N. Riseth, "Optim: A mathematical optimization package for Julia," *Journal of Open Source Software*, vol. 3, no. 24, 2018.

[29] S. A. Broda and M. S. Paolella, "ARCHModels.jl: Estimating ARCH models in Julia," 2020.

[30] A. Ghalanos, "rugarch: Univariate GARCH Models. R package version 1.4-2 [Online]. Available: https://CRAN.R-project.org/package=rugarch," 2020.