NTNU
Kunnskap for en bedre verden

Department of Mathematics

TMA4500 - Industrial Mathematics, Specialization Project

Stochastic Volatility Models

Author:
Thomas Torkildsen

Abstract

For financial time series, the volatility is mainly modeled using stochastic volatility (SV) or generalized autoregressive conditional heteroscedastic (GARCH) models. Due to their simplicity, GARCH models are used much more frequently than SV models. Even though SV models often outperform GARCH models. This paper uses the **TMB** library to fit a SV model. By parametric bootstrap, it is shown that the parameters of the SV model are estimated well for most choices of parameter values. Finally, the fit of a SV model is tested on two example time series and compared to the fit of a GARCH(1,1) model. The results show for both examples that the SV model is the best. Although implementing a SV model is more complicated than a GARCH model, this will likely make more people consider using SV models.

September, 2022

Contents

List of Figures

List of Tables

1 Introduction

When analyzing financial time series, the change in asset price can often be considered to follow a random walk and are difficult to forecast precisely. However, volatility usually has properties that can be explained. When analyzing the volatility of a time series, stochastic volatility (SV) models and generalized autoregressive conditional heteroscedastic (GARCH) models are most used. Due to the simplicity of the model, GARCH models are used much more frequently than SV models. Martino et al., 2011 suggest that since SV models include two error terms, while GARCH models only have one, a GARCH model is less flexible than a SV model.

This paper consists of two parts. In the first part, a SV model with a logarithmic AR(1) volatility term is implemented and tested. Due to the model complexity, it has to be implemented using the open source R library; template model builder (**TMB**). With the **TMB** library, maximum likelihood (ML) estimates of model parameters can be approximated for complex latent variable models. Template model builder calculates random effects and parameters using Laplace approximations and automatic differentiation. All relevant theory about TMB can be found in sections 2.4 -2.6. The sections 3.1 and 3.2 describe the SV model and its implementation in detail.

Next, the reliability of the implemented model is evaluated. The model is evaluated using parametric bootstrap and measuring the distance between estimated and actual model parameters. Parametric bootstrap for a SV model is elaborated in section 2.8. In section 4.1, the results of the parametric bootstrap are presented for several choices of actual parameter values. Finally, the results are discussed in section 5.1.

In the second part of the paper, the SV model, from the first part, and a GARCH(1,1) model are fitted to two different financial time series (3.7). The best model is determined by using cross-validation (2.7) to compute an evaluation metric described in section 3.5. Finally, the results are presented in section 4.2 and discussed in section 5.2.

2 Theory

This section contain all relevant theory for needed to understand the implementation in section 3.

2.1 General definitions

Stationary time series:

In Brockwell and Davis, 2016a a time series is considered weakly stationary if the expected value $E[y_t] = \mu_t = \mu$ is equal for all t, and $\gamma(s,t) = \gamma(s+h, t+h)$ for all s, t and h, where

γ is the auto-correlation function and is defined as $\gamma(s,t) = E[(y_s - \mu_s)(y_t - \mu_t)]$

In this paper a weakly stationary time series is considered stationary.

White Noise:

In Brockwell and Davis, 2016a a time series is considered white noise if it is a stationary process with zero auto-correlation. In other words, if for all $t \in \mathbb{Z}$, then $a_t \overset{indep.}{\sim} N(0,b)$, then $\{a_t\}$ is white noise and can be written on the form $a_t \sim WN(0,b)$.

2.2 ARMA models

ARMA is short for autoregressive moving average and is a model used to describe the movement of an observed time series. In Brockwell and Davis, 2016a an ARMA(p,q) process is defined as:

$\{y_t\}$ is an ARMA(p,q) process if $\{y_t\}$ is stationary and if for every $t \in \mathbb{Z}$,

$$y_t - \phi_1 y_{t-1} - ... - \phi_p y_{t-p} = \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} \tag{1}$$

where $\{y_t\}$ is the observed value of the time series, $\{\epsilon_t\} \sim WN(0, \sigma^2)$ and the polynomials $(1 - \phi_1 z - ... - \phi_p z^p)$ and $1 + \theta_1 z + ... + \theta_q z^q$ have no common factors. An ARMA(p,q) process also has to be causal and invertible.

In Brockwell and Davis, 2016a an ARMA(p,q) process is considered invertible if for all $|z| \leq 1$, then

$$\theta(z) = 1 + \theta_1 z + ... + \theta_q z^q \neq 0 \tag{2}$$

Likewise, an ARMA(p,q) process is considered cousal if for all $|z| \leq 1$, then

$$\phi(z) = 1 - \phi_1 z - \phi_p z^p \neq 0. \tag{3}$$

An AR(1) (ARMA(1,0)) process has the form

$$y_t - \phi y_{t-1} = \epsilon_t \tag{4}$$

where, in order to be causal and invertible, the ϕ is constrained by $|\phi| < 1$

An ARMA(1,1) process has the form

$$y_t - \phi y_{t-1} = \epsilon_t + \theta \epsilon_{t-1} \tag{5}$$

with the boundaries $|\phi| < 1$ and $|\theta| < 1$.

2.3 Latent variable models

In short, a latent variable model is a model that includes latent variables to explain the behavior of observed variables in a statistical model. A latent variable is usually constructed assuming conditional independence between the observed variables. Hence, the likelihood function of a latent variable model has the following form

$$f(\boldsymbol{y}, \boldsymbol{x}) = f(\boldsymbol{x}) \prod_{i=1}^{n} f(y_i|\boldsymbol{x}) \tag{6}$$

where $\boldsymbol{x}$ is a considered a vector of latent variables and $\boldsymbol{y}$ the observed variables. By construction $f(y_i|\boldsymbol{x})$ are independent for all $i \in \{1, ..., n\}$. More on latent variable models and their construction in Bishop, 1998.

2.4 Laplace approximation

In this paper, the R package TMB is used to fit a stochastic volatility model as explained in section 3. The TMB package uses Laplace Approximation for parameter estimation in complicated models. In TMB, Laplace approximations are used as follows (Kristensen et al., 2016).

Consider a negative joint log-likelihood function on the form $f(\mathbf{u}, \theta; \mathbf{y})$, where θ is a vector of unknown model parameters, $\mathbf{u}$ random variables and $\mathbf{y} = (y_1, ...y_n)$ the observed time series. The maximum likelihood estimate for θ maximizes the marginal likelihood as follows

$$L(\theta; \mathbf{y})) = \int_{\mathbb{R}^n} \exp(-f(\mathbf{u}, \theta; \mathbf{y}))d\mathbf{u} \tag{7}$$

with respect to θ. Often, it is difficult to integrate out the random effects as in (7). Laplace approximation can be used to approximate the integral above. The Laplace approximation uses a Taylor expansion to approximate a normal distribution as follows

$$L(\theta; \mathbf{y})) \approx \int_{\mathbb{R}^n} \exp(-f(\hat{\mathbf{u}}(\theta), \theta; \mathbf{y}) - \frac{1}{2}(\mathbf{u} - \hat{\mathbf{u}}(\theta))^T H(\theta)(\mathbf{u} - \hat{\mathbf{u}}(\theta)))d\mathbf{u} \equiv L^*(\theta; \mathbf{y}) \tag{8}$$

where $\hat{\mathbf{u}}(\theta)$ is

$$\hat{\mathbf{u}}(\theta) = \arg\min_{\mathbf{u}} f(\mathbf{u}, \theta; \mathbf{y}) \tag{9}$$

and $H(\theta)$ is the Hessian of $f(u, \theta; \mathbf{y})$ differentiated with respect to $\mathbf{u}$ and evaluated at $\hat{\mathbf{u}}(\theta)$ as follows

$$H(\theta) = f''_{\mathbf{uu}}(\hat{\mathbf{u}}(\theta), \theta; \mathbf{y}). \tag{10}$$

Now, equation (8) can be rewritten as

$$L^*(\theta; \mathbf{y}) = \exp(-f(\hat{\mathbf{u}}(\theta), \theta; \mathbf{y})) \int_{\mathbb{R}^n} \exp(-\frac{1}{2}(\mathbf{u} - \hat{\mathbf{u}}(\theta))^T H(\theta)(\mathbf{u} - \hat{\mathbf{u}}(\theta)))d\mathbf{u} \tag{11}$$

Since a multivariate normal distribution with expectation μ and covariance matrix Σ has a PDF on the form

$$f_{\mathbf{x}}(\mathbf{x}; \mu, \Sigma) = (2\pi)^{-\frac{n}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu))$$

with the following integral

$$\int_{\mathbb{R}} f_{\mathbf{x}}(\mathbf{x}; \mu, \Sigma) d\mathbf{x} = 1. \tag{12}$$

Using equation (12), it can easily be proven that the solution of the Laplace approximation has the solution:

$$L^*(\theta; \mathbf{y}) = (2\pi)^{\frac{n}{2}} \det(H(\theta))^{-\frac{1}{2}} \exp(-f(\hat{\mathbf{u}}(\theta), \theta; \mathbf{y})) \tag{13}$$

The Laplace approximation requires regularity conditions on the joint negative log-likelihood functions; e.g., the minimizer of $f(\mathbf{u}, \theta; \mathbf{y})$ with respect to θ has to be unique.

The final estimate of θ minimizes:

$$-\log L^*(\theta; \mathbf{y}) = -\frac{n}{2}\log(2\pi) + \frac{1}{2}\log\det(H(\theta)) + f(\hat{\mathbf{u}}(\theta), \theta; \mathbf{y}) \tag{14}$$

Further, the uncertainty of the estimate $\hat{\theta}$ of any differentiable function of the estimate $\phi(\hat{\theta})$ can be found by using the δ-method. Which in Kristensen et al., 2016 is defined by:

$$\text{var}(\phi(\hat{\theta})) = -\phi'(\hat{\theta})(\nabla^2 \log L^*(\hat{\theta}))^{-1}\phi'(\hat{\theta})^T \tag{15}$$

When computing (14), efficient and accurate computations of derivatives are needed. How this is done in the TMB package is explained in section 2.5

2.5 Automatic differentiation

Automatic differentiation:

When computing the Laplace approximation of a marginal likelihood function, the ability to compute quick and precise derivatives is essential. There are several ways to compute derivatives on a computer, e.g., symbolic, numerical, and automatic differentiation. In Baydin et al., 2015, the differences between the corresponding methods are explained as follows. Symbolic differentiation is a method for calculating the analytical solution to a derivative for all input values of the function, not only in a given point and resulting in exact computations of the derivative. The problem with symbolic differentiation is that a function's derivative might be exponentially longer than the function, resulting in long run times.

One way to overcome the problems of expression swell is by using numerical differentiation. Numerical differentiation uses the original function, $\mathbf{f}$, to estimate its derivative. The simplest method for numerical differentiation is the forward difference.

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} + O(h^2) \tag{16}$$

where $h > 0$ is a small step size and $\mathbf{e}_i$ is a unit distance in direction i and $O(h^2)$ is the truncation error. Numerical differentiation also produces a round-off error, which is the difference between the true estimated value and its computed approximation. The total error is the sum of the truncation and round-off error and is illustrated as a function of the step size (h) in figure 1. In the same figure, the error for the more complicated center difference is also plotted. The round-off error is the dominating error for small step sizes for both methods. No matter how complex a numerical differentiation method is, a round-off error always occurs.

By using automatic differentiation, the deficiencies with numerical and symbolic differentiation are omitted (Baydin et al., 2015). Automatic Differentiation (AD) quickly computes a precise

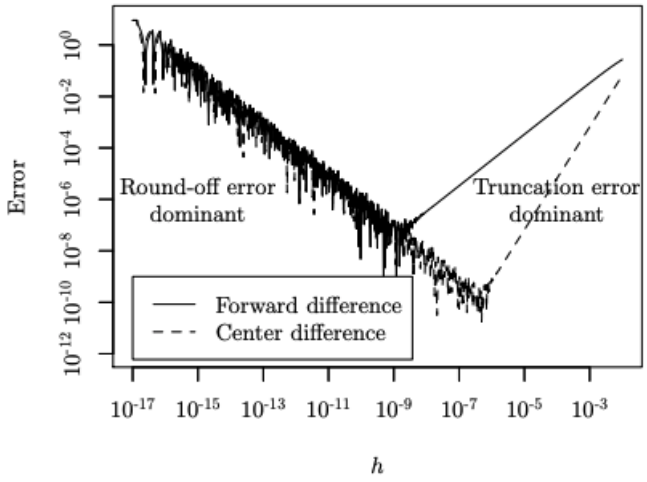

Figure 1: A plot of the total error for different grid sizes (h). The dotted line is for the center difference, and the forward difference is for the solid line. The Total error is the sum of the truncation error and round-off error. The optimal grid size is where the total error is smallest. The figure is from Baydin et al., 2015

Forward Evaluation Trace			Forward Evaluation Derivative Trace	
1.	$v_0 = x_1$	$= 0.5$	$\dot{v}_0 = \dot{x}_1$	$= 1$
2.	$v_1 = x_2$	$= 0.1$	$\dot{v}_1 = \dot{x}_2$	$= 0$
3.	$v_2 = e^{v_0}$	$= e^{0.5}$	$\dot{v}_2 = e^{v_0}\dot{v}_0$	$= e^{0.5}$
4.	$v_3 = v_0 v_1$	$= 0.05$	$\dot{v}_3 = \dot{v}_0 v_1 + v_0 \dot{v}_1$	$= 0.5$
5.	$v_4 = \cos v_1$	$= \cos 0.1$	$\dot{v}_4 = -\sin(v_1)\dot{v}_1$	$= -\sin(0.1)0$
6.	$v_5 = v_2 + v_3$	$= 1.65 + 0.05$	$\dot{v}_5 = \dot{v}_2 + \dot{v}_3$	$= 1.65 + 0.5$
7.	$v_6 = v_5 - v_4$	$= 1.65 + 0.05$	$\dot{v}_6 = \dot{v}_5 - \dot{v}_4$	$= 2.15 - 0$
8.	$f = v_6$	$= 1.70$	$\dot{f} = \dot{v}_6$	$= 2.15$

Table 1: An example of forward mode AD, with example function $f(x_1, x_2) = e^{x_1} + x_1 x_2 - \sin x_2$, evaluated at $(x_1, x_2) = (0.5, 0.1)$. The derivative $\frac{\partial f}{\partial x_1}(0.5, 0.1)$ is computed. On the left is the forward prime trace and on the right is the forward derivative trace.

derivative at a given point $\boldsymbol{x}$ by using symbolic rules of differentiation combined with numerical input values.

There are several ways to do automatic differentiation. In **TMB**, automatic differentiation is done through the external library **CppAD**, using operator overloading. Operator overloading is the same as abstracting away the derivative computation. Automatic differentiation uses the fact that every function can be split into a finite number of elementary operations. For every operation, the corresponding derivative is known. For example, the derivative of the "+" operation is

$$\frac{\partial(u+v)}{\partial x} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial x}$$

To illustrate how automatic differentiation works, consider the example function $f(x_1, x_2) = e^{x_1} + x_1 x_2 - \cos x_2$. In table 1, the order in which elementary operations are done can be seen on the left-hand side. Similarly, the order of the operations can be visualized in figure 2. The arrows indicate which variables are used in the calculation of the next.

Forward mode is when the partial derivative is calculated in the same order as the right-hand side in table 2. For each run, one partial derivative is estimated. In the example this is $\frac{\partial f}{\partial x_1}|_{x=(0.5,0.1)}$. So for $f : \mathbb{R}^n \to \mathbb{R}^m$ n runs are required to compute the Jacobean.

The reversed mode stores the direction of the partial derivative is calculated in the opposite order as in figure 2. In this case, the derivatives are calculated backward. This is done in a two-part

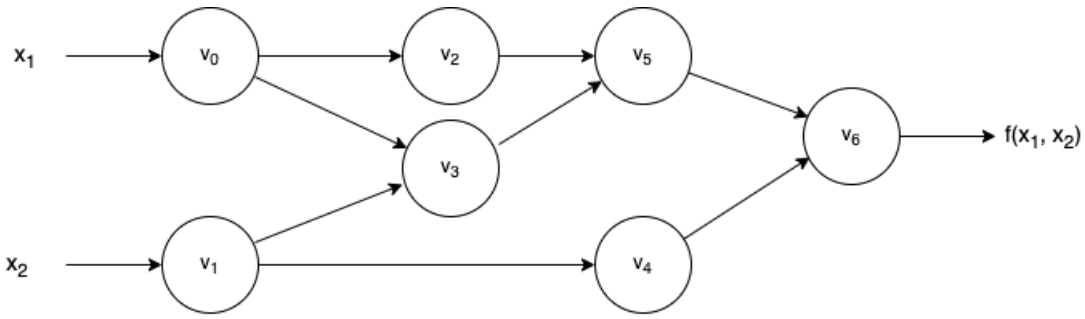

Figure 2: Flowchart of elementary operations on the example function $f(x_1, x_2) = e^{x_1} + x_1 x_2 - \cos x_2$ See the left hand side of table 1 for a more depth on the flow of operations.

process. First, a forward pass is done, and the intermediate variables are estimated and stored with the dependencies between them. Then, the partial derivatives of the output are computed with respect to the intermediate derivatives as follows.

$$\bar{v}_i = \frac{\partial f}{\partial v_i} = \sum_{j:\text{child of i}} \bar{v}_j \frac{\partial v_j}{\partial v_i} \tag{17}$$

where $\bar{f} = \bar{v}_6 = 1$. Finally, after iterating backward through the full jacobian is $(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}) = (\bar{v}_0, \bar{v}_1)$

In TMB, the computation of the Hessian is done by (10) with a combination of backward and forward modes.

2.6 TMB

Template Model Builder (TMB) is an open-source R package used to implement complex latent variable models like the AD model Builder package. Template Model Builder estimates random effects and parameters, finding the Laplace approximation of the likelihood (see 2.4). The Hessian used in the Laplace approximation equation 14 is computed by automatic differentiation (see section 2.5).

To use TMB, the user writes the negative logarithm of the joint likelihood as a function of θ and u in a C++ file.

In the R file, AD function objects are constructed using the function **MakeADFun** from the **TMB** library. The functions input values are the observations ($\mathbf{y}$), initial values for the random variables (θ, u), and a string specifying which parameters (u) are to be integrated out of the negative log-likelihood function.

To find the ML estimates of the specified model in R, the function **nlminb** is used. The function takes the AD function objects returned by **MakeADFun** as input and returns the ML estimates of the model parameters (θ). In **nlminb**, the ML estimates are calculated using Laplace approximation combined with AD, as described in 2.4 and 2.5. (Kristensen et al., 2016)

2.7 K-fold Validation on Time Series

Since the observations of a time series are dependent, regular K-fold cross-validation is unsuitable for cross-validation. An alternative is rolling cross-validation, as done in (Pedregosa et al., 2011). Rolling cross-validation fixes the time dependence problem by testing unseen (out-of-sample) data. Figure 3 is an example of rolling cross-validation with 3-folds.

To explain how rolling cross-validation works, consider a time series with 100 observations evaluated

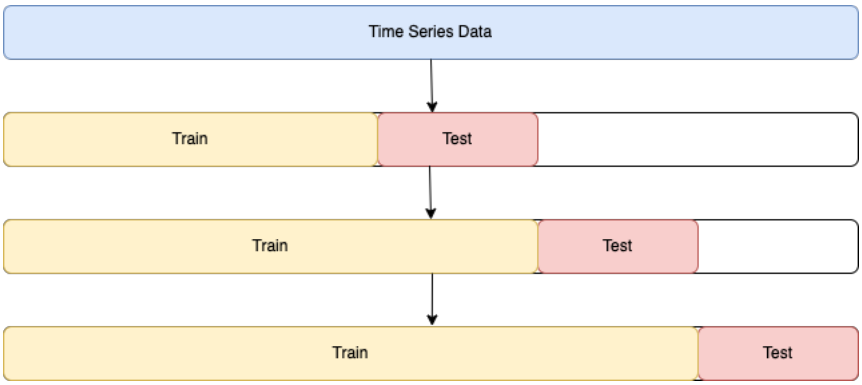

Figure 3: Illustration of 3-fold rolling cross-validation on a time series. The order of each test/train-split follow the arrows.

twice.

First, a time series is split into three parts where the first part is bigger than the other and, let's say that the first has 50 observations, the second has the next 25, and the third has the last 25 observations. as follows:

$$\{Y_1, ..., Y_{100}\} \rightarrow \{\{Y_1, ..., Y_{50}\}_1, \{Y_{51}, ..., Y_{75}\}_2, \{Y_{76}, ..., Y_{100}\}_3\}$$

Then a model is fitted on $\{Y_1, ..., Y_{50}\}_1$ and evaluated on $\{Y_{51}, ..., Y_{75}\}_2$. In this paper, the model is evaluated as described in section 3.5, where the evaluation value is saved for each iteration. The final validation model is fitted to the combination of the first two parts of the time series, $\{\{Y_1, ..., Y_{50}\}_1, \{Y_{51}, ..., Y_{75}\}_2\}$. Then evaluated on the last $\{Y_{76}, ..., Y_{100}\}_3$. In Bergmeir and Benıtez, 2012, cross-validation on time series is explained in more detail.

2.8 Parametric bootstrap

To evaluate the performance of the model parameter estimations found using TMB, a version of Parametric bootstrapping is used. Parametric bootstrap can be used to measure the uncertainty of the parameter estimates.

A condition for parametric bootstrap is that a model must fit the data vel so that its residuals can be considered random noise. If so, then parametric bootstrap can be used as follows.

First, the ML parameters of the model are estimated. After the ML parameters are estimated, the residuals are found. These residuals are then drawn randomly and added back into the model, constructing a new data set. Then the ML parameters are estimated and stored. This procedure is repeated until one ends up with a distribution of ML estimates.

For the case of estimating known parameters of a time series, it does not make sense to draw samples of an already fitted model—instead, a new Monte Carlo simulation of the SV model for each run. Then TMB is used to compute and store the ML estimates. See 1 for a step-by-step illustration of how parametric bootstrap is done for the parameter estimation in this paper.

Algorithm 1 Parametric Bootstrap

1: **for** b=1,...,B **do**
2: $\mathbf{S}[b] \leftarrow$ simulateSVM$(n, \theta, \phi, \mu, \sigma)$ ▷ Function that simulates an instance of observed values from the SV model described in (18) for a given input.
3: $\hat{\mathbf{\Theta}}[b] \leftarrow$ estimateParametes$(S[b])$ ▷ Parameter estimation by same method described in 3.6
4: **end for**

The vector of maximum likelihood estimates is used for further analysis; see 3.6.

3 Data and Model

When comparing the stochastic volatility model defined in subsection 3.1, it is essential to know how well the model performs on a data set with known parameters. ML estimates of the model parameters are evaluated as described in 3.6 by using bootstrap, 2.8. After the SV model is implemented as in 3.2, it is tested as in 3.6 and then compared to GARCH(1,1) (21) model by looking at the data described in 3.7.

3.1 The Stochastic Volatility model

In this paper, the stochastic volatility (SV) model is defined by:

$$\begin{aligned} y_t &= y_{t-1} + \theta + \exp(\frac{x_t}{2})u_t \\ x_t &= \mu + \phi(x_{t-1} - \mu) + v_t \end{aligned} \tag{18}$$

Where for all $t \in \mathbb{Z}$, y_t can be considered a random walk with an expected change in observed value, θ, each time step, t. Since $u_t \overset{i.i.d}{\sim} N(0,1)$, x_t is logarithm of the variance at time step t. The variable x_t is an autoregressive process with one term (AR(1)), where μ and ϕ are constants and $v_t \overset{i.i.d}{\sim} N(0, \sigma^2)$.

Assuming y_1 is known, the joint density function is on the form

$$\begin{aligned} f(y_2, ..., y_n, x_1, ..., x_n | y_1) &= f(x_1, ..., x_n) f(y_2, ..., y_n | x_1, ..., x_n) \\ &= f(x_1) \prod_{i=2}^{n} f(x_i | x_{i-1}) \prod_{i=2}^{n} f(y_i | y_{i-1}, x_i) \end{aligned} \tag{19}$$

Where $x_1 \sim N(\mu, \frac{\sigma^2}{1-\phi^2})$ is the marginal distribution of x_t for an arbitrary t, $x_i | x_{i-1} \sim N(\mu + \phi(x_{i-1} - \mu), \sigma^2)$ and $y_i | y_{i-1}, x_i \sim N(y_i + \theta, e^{x_t})$

From the definition of an ARMA-processes in section 2.2 it follows that $-1 < \phi < 1$ for a stationary AR(1) model.

In section 3.2, the procedure of finding the maximum likelihood estimates for the SV model is explained.

3.2 SV Model Implementation

To fit a stochastic volatility model to a time series, the model parameters need to be estimated. This is done by finding the parameters that maximize the Laplace approximation of the model marginal likelihood function

$$L(\theta, \mu, \phi, \sigma^2 | \mathbf{y}) = \int \cdots \int \exp(-g(y_2, ..., y_n, x_1, ..., x_n | y_1)) dx_1 \cdots dx_n, \tag{20}$$

with respect to the parameters. Here, $g(y_2, ..., y_n, x_1, ..., x_n | y_1) = -\log f(y_2, ..., y_n, x_1, ..., x_n | y_1)$ is the negative logarithm of equation (19). The parameters that maximize equation (20) are approximated using the **TMB** package in R. The iteration scheme, when using the package, becomes:

1. find $\hat{\mathbf{x}}(\theta, \mu, \phi, \sigma^2) = \arg\min_{\mathbf{x}} g(x)$
2. calculate $H(\theta, \mu, \phi, \sigma^2) = g_{\mathbf{xx}}(\hat{\mathbf{x}}, \theta, \mu, \phi, \sigma^2)$
3. $-\log L^*(\theta, \mu, \phi, \sigma^2 | \mathbf{y}) = g - \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|H(\theta, \mu, \phi, \sigma^2)|$
4. find $(\hat{\theta}, \hat{\mu}, \hat{\phi}, \hat{\sigma}^2) = \underset{\theta, \mu, \phi, \sigma^2}{\arg\min}(-\log L^*(\theta, \mu, \phi, \sigma^2 | \mathbf{y}))$

All the steps above are done numerically using the **TMB** package in R. However, the negative log-likelihood function, g, had to be implemented manually in a C++ file. (see Appendix (section 7) for the implementation in of g). In the first and forth step, $\hat{\mathbf{x}}$ and $(\hat{\theta}, \hat{\mu}, \hat{\phi}, \hat{\sigma}^2)$ are found by numerical optimization. The Hessian in the second step is Computed with AD. The estimated standard deviations of the parameter estimates are also computed automatically using **TMB**.

See section 2.4, 2.5, and 2.6 for more details on Laplace approximation, Automatic differentiation, and **TMB**.

3.3 GARCH models

GARCH is a model for predicting dependencies in changing volatility. GARCH is short for generalized autoregressive conditional heteroskedasticity and was first defined in the paper Bollerslev, 1986. For a GARCH(p,q)-process, the magnitude of the volatility is considered to have the following form

$$\begin{aligned} y_t &= y_{t-1} + \theta + \sigma_t u_t \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2, \end{aligned} \tag{21}$$

where the general volatility of the observed time series is on the form $\epsilon_t = \sigma_t u_t$. where $u_t \sim WN(0,1)$ and σ_t^2 follows the form defined above. Further model assumptions are; $\alpha_i \geq 0$, $\beta_i \geq 0$ and $\omega > 0$ and $\epsilon_t^2 \sim N(0, \sigma_t^2)$.

A GARCH(p,q) process can be rewritten on the form of an ARMA(p,q) process by defining (Brockwell and Davis, 2016a)

$$v_t = \epsilon_t^2 - E_{t-1}(\epsilon_t^2) = \epsilon_t^2 - \sigma_t^2, \tag{22}$$

and then insert equation (22) into equation (21) to obtain

$$\epsilon_t^2 - v_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i (\epsilon_{t-i}^2 - v_{t-i}). \tag{23}$$

This can, in turn, be rewritten as

$$\epsilon_t^2 = \omega + \sum_{i=1}^{\max(p,q)} (\alpha_i + \beta_i)\epsilon_{t-i}^2 + v_t - \sum_{i=1}^{p} \beta_i v_{t-i}. \tag{24}$$

Since $\{v_t\}$ become uncorrelated white noise, $\{\epsilon_t^2\} \sim \text{ARMA}(\max(p,q), p)$. Now a GARCH(1,1) can be written as

$$\epsilon_t^2 = \omega + (\alpha + \beta)\epsilon_{t-1}^2 + v_t - \beta v_{t-1} \tag{25}$$

with the GARCH-conditions $\alpha \geq 0$ $\beta \geq 0$ and $\omega > 0$. Since the GARCH(1,1)-process also is a ARMA(1,1) process also have the constraints $|\alpha + \beta| < 1$ and $|\beta| < 1$. This gives the following restrictions a GARCH(1,1) process

$$\alpha + \beta < 1, \quad \beta < 1 \tag{26}$$

A random walk with a GARCH(1,1) volatility term can be written in the following form.

$$\begin{aligned} y_t &= y_{t-1} + \theta + \epsilon_t = y_{t-1} + \theta + \sigma_t u_t \\ \sigma_t^2 &= \omega + \alpha \epsilon_{t-1}^2 + \beta \sigma_{t-1}^2 \end{aligned} \tag{27}$$

where once again y_t $t \in \{1, 2, ..., n\}$ is considered a random walk with an expected increase θ each time step and a volatility term. $u_t \overset{i.i.d}{\sim} N(0, 1)$ is white noise so that the volatility term follows a GARCH(1,1).

The joint density for function for a time series with observations $\{y_2, ..., y_n\}$, can for a given y_1, σ_1^2 and ϵ_1 be defined as

$$f(y_2, ..., y_n | y_1, \sigma_1^2) = \prod_{i=2}^{n} f(y_i | y_{i-1}, ..., y_1, \sigma_1^2) \tag{28}$$

where $y_i | y_{i-1}, ..., y_1, \sigma_1^2 \sim N(y_{i-1} + \theta, \sigma_i^2)$ and σ_i^2 is known given and time step $i - 1$. Since the product of gaussian PDFs is gaussian, equation (28) is also gaussian. Thus, the parameter estimates of the GARCH(1,1) can easily be found using maximum likelihood directly on (28) (Brockwell and Davis, 2016b). The model parameters' standard deviation can be estimated using the fisher information matrix.

3.4 Comparing Marginal Variances

The GARCH and SV model, defined in section 3.3 and 3.1, are different in many ways. The random walk for the two models can be written as

$$\begin{aligned} y_t &= y_{t-1} + \theta^{SVM} + \epsilon_t^{SVM} \\ y_t &= y_{t-1} + \theta^{GARCH} + \epsilon_t^{GARCH} \end{aligned} \tag{29}$$

where $t \in \{1, \ldots, n\}$ is the time step, θ^m and ϵ_t^m is the drift and volatility of model m. An obvious parameter to compare is θ^m, as this is the expected change in the observed value for each time step. For both models, ϵ_t can be considered normally distributed with some variance V_t. For the SVM $V_t^{SVM} = e^{x_t}$ and for GARCH $V_t^{GARCH} = \sigma_t^2$.

Since the expectation of the marginal log-volatility of the SV model has the distribution $x_t \sim N(\mu, \frac{\sigma^2}{1-\phi^2})$, then $\exp(x_t) \sim \log N(\mu, \frac{\sigma^2}{1-\phi^2})$. Therefore, the expected marginal variance of the SVM is

$$V_t^{SVM} \equiv E[\exp(x_t)] = \exp(\mu + \frac{\sigma^2}{2(1-\phi^2)}). \tag{30}$$

Likewise, in (Williams, 2011) the expected marginal variance for the GARCH(1,1) is be proven to be

$$V_t^{GARCH} \equiv E[\sigma_t^2] = \frac{\omega}{1 - \alpha - \beta}. \tag{31}$$

So that the expected marginal variance of a SVM and a GARCH(1,1) model, fitted to the same time series, can be compared by comparing (30) and (31).

3.5 Cross-Validation Scores

When looking at the performance of a volatility model, it is natural to not only look at the forecast of the observed value, y_{n+1}. Looking at the forecast alone is not interesting because the expected value of a random walk is independent of the volatility term.

$$\hat{y}_{n+1} = E[y_{n+1} | y_n, y_{n-1}, ...] = E[y_n + \hat{\theta} + \epsilon_n | y_n, y_{n-1}, ...] = y_n + \hat{\theta}. \tag{32}$$

For this reason, traditional metrics such as the root mean square error is useless and cannot explain how well the volatility is predicted.

Since the volatility term can be considered a latent variable in each model (volatility is not observed), the mean square error of predicted volatility cannot be calculated. In other words, the actual volatility is unknown, so the distance from the actual volatility to the estimated one cannot be quantified.

This problem can be solved by looking at the marginal likelihood function $f(y_1, ..., y_n; \hat{\Theta})$. The marginal contains information about the latent variables, the model performance for a given MLE $\hat{\Theta}$ can therefore be quantified by looking at the performance of the conditional likelihood of the validation set $y_{n+1}, ..., y_{n+h}$. The conditional likelihood of the validation set for given parameters $\hat{\Theta}$, is calculated as follows

$$f(y_{n+1}, ..., y_{n+h}|y_1, ..., y_n; \hat{\Theta}) = \frac{f(y_1, ..., y_n, y_{n+1}, ..., y_{n+h}; \hat{\Theta})}{f(y_1, ..., y_n|; \hat{\Theta})} \tag{33}$$

where $\hat{\Theta}$ is the ML estimates for the marginal likelihood function $f(y_1, ..., y_n|; \Theta)$. The marginal likelihood function for the SV model is found with its corresponding ML estimators by using TMB as described in section 3.2. For the GARCH(1,1), its ML estimators are the parameters that maximize (28). The likelihood $y_1, ..., y_{n+h}$ is calculated by inserting the ML estimates, $\hat{\Theta}$, into the joint likelihood model.

The metric used for evaluating the out-of-sample performance of each respective volatility model is the logarithm of (33), defined as

$$l_m^{n,h}(\hat{\Theta}_m^n) \equiv \log f_m(y_{n+1}, ..., y_{n+h}|y_1, ..., y_n; \hat{\Theta}_m^n) \tag{34}$$

where $l_m^{n,h}$ means the conditional log likelihood of model m on on observation $y_{n+1}, ..., y_{n+h}$.$\hat{\Theta}_m^n$ is the ML estimates of model m on $y_1, ..., y_n$. By itself equation 34) does not give an intuitive understanding of the performance. However, two models with similar likelihoods can be compared. The best model for each evaluation interval $n+1, ..., n+h$ is the model with the highest conditional log-likelihood. Therefore the models can be directly compared by looking at

$$M(n+1, n+h) \equiv l_{SVM}^{n,h}(\hat{\Theta}_{SVM}^n) - l_{GARCH}^{n,h}(\hat{\Theta}_{GARCH}^n) \tag{35}$$

$M(n+1, n+h)$ is the difference between the conditional log-likelihood of the SVM and GARCH(1,1) models, evaluated at the out-of-sample time steps $n+1, n+2, ..., n+h$. If $M(n+1, n+h) > 0$ indicates that SVM outperforms GARCH(1,1) on the out-of-sample interval and $M(n+1, n+h) < 0$, that GARCH(1,1) outperforms SVM on the out of sample interval. To ensure the outperformance of a model is not random, M is calculated several times using the rolling cross-validation described in section 2.7. Finally, the overall best model is the model with the highest score

$$SCORE \equiv \sum_{i=1}^{N} M(n + (i-1)h, n + ih) \tag{36}$$

where N is the number of rolling validation sets. If $SCORE > 0$, SVM is considered the best model, and if $SCORE < 0$, then GARCH(1,1) is the best model. $SCORE$ can be split into the difference between the sum of each conditional model models as follows

$$SCORE = L_{SVM} - L_{GARCH} \tag{37}$$

Where L_m can be considered the total log-likelihood for model m and is defined as

$$L_m = \sum_{i=0}^{N-1} l_m^{n+ih,h}(\hat{\Theta}_m^{n+ih}) \tag{38}$$

The $SCORE$, L_{SVM} and L_{GARCH} for two example models are presented in 4.2.

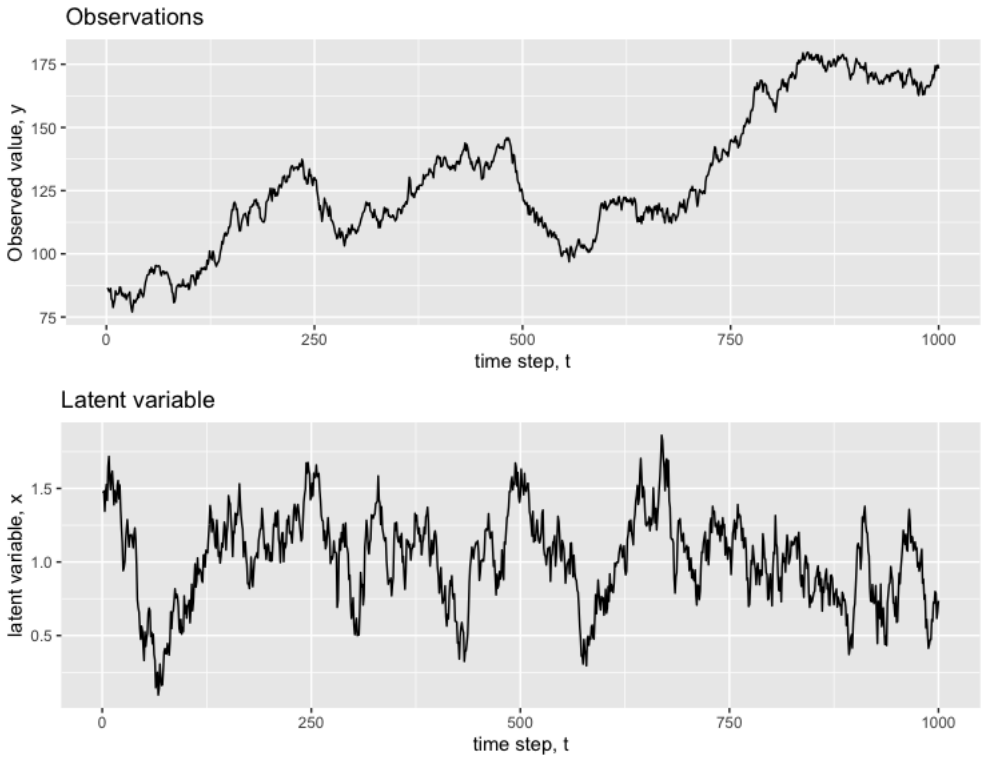

Figure 4: A simulation of a stochastic volatility model defined in 3.1. Along the x-axis of each graph is the time step. The upper graph contains the observed values $\{y_t\}$. The lower graph is the latent variable and the logarithm of the volatility, $\{x_t\}$. The model parameters are $\theta = 0.1$, $\mu = 1$, $\phi = 0.95$, $\sigma = 0.1$.

3.6 Parameter estimation of the SVM

Stochastic volatility models are challenging to estimate compared to GARCH models. The reason is that a Stochastic volatility model´s parameters cannot be directly estimated through maximum likelihood. For a stochastic volatility model, the parameter values are calculated by maximizing the marginal likelihood of the observed time series with respect to the model parameters, as in the iteration 3.2. The marginal likelihood is found by integrating out the random effect by using TMB (section 3.2). As the marginal log-likelihood is found using Laplace approximations, a slight deviation from the actual model is likely.

The accuracy of ML parameter estimates will be evaluated graphically. This is done using parametric bootstrap, as described by algorithm 1. From the parametric bootstrap, a vector of MLEs is returned for each parameter. Interesting quantities are looked at for these vectors, such as the 5%, 95%, and 50% quantiles. Since the actual parameter values are known in advance, several metrics can be used to quantify the accuracy of each ML estimate. However, in this paper, it will be sufficient to evaluate the performance visually.

When looking at a stochastic volatility model, it is expected to be easier to estimate the model parameters for some choice of parameters relative to others. For example, it is likely that a process with a high correlation coefficient, ϕ, is easier to estimate than a model where $\phi = 0$. Likewise, the same is expected for a time series with many observations relative to one with few observations. The process with many observations will probably have more accurate parameter estimations than the one with few observations. White noise might not be considered random for models with few observations. It makes sense that the probability of detecting a pattern in white noise is reduced as the length of the time series increases.

Figure 4 shows a simulation of a stochastic volatility model. The model is simulated using the parameters $\theta = 0.1$, $\mu = 1$, $\phi = 0.95$, $\sigma = 0.1$. In the upper plot are the observed time series and

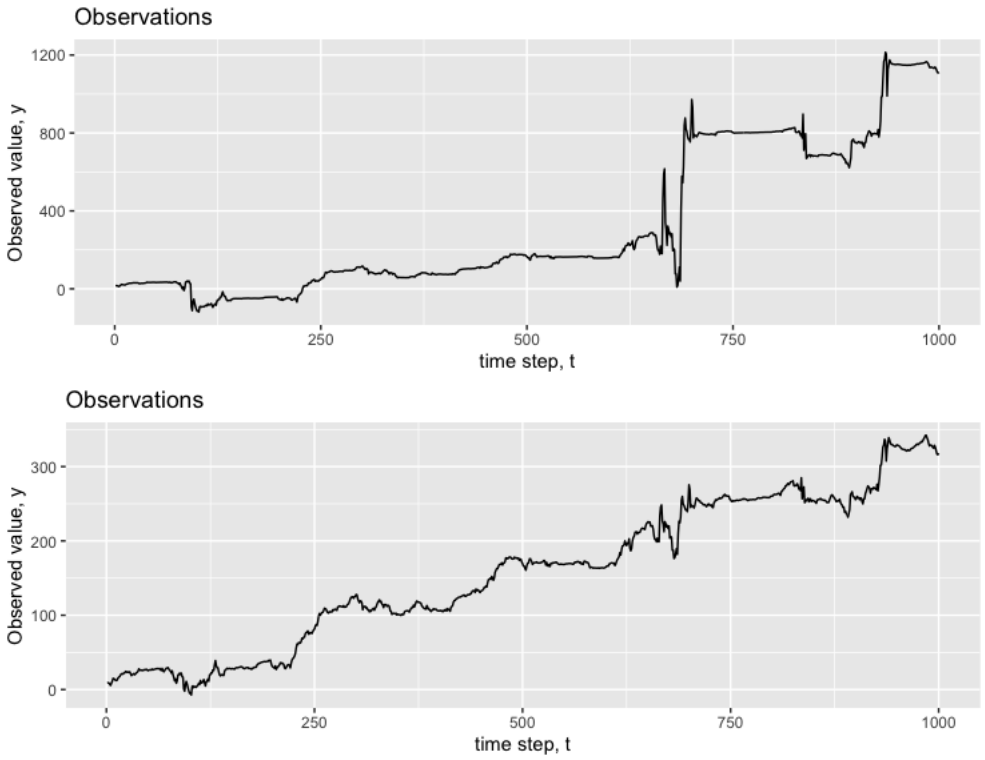

Figure 5: This figure contains simulations of stochastic volatility processes. models with $\theta = 0.1$, $\mu = 1$, $\phi = 0.95$, they are also simulated from the same seed. However, $\sigma = 1$ for the upper plot and $\sigma = 0.5$ for the lower.

the unobserved log-volatility time series at the bottom. The model looks like a time series that could be from the real world.

Since the logarithm of the volatility term follows an AR process, it is natural to assume that the observed time series quickly becomes unreasonably volatile for sudden big instances $\{x\}$. Although high volatility occurs at some frequency for real-world time series, they do not happen often. Therefore huge values of σ is likely not applicable to real-world time series. Two examples of SV-processes with high values of σ are shown in figure 5. The upper figure illustrates the observations of an objective function with $\sigma = 1$, whereas the plot below has $\sigma = 0.5$. The difference between the figures are huge, and indicates that the parameter values need to be considered carefully.

Another thing that might lead to difficulties in parameter estimation is a small marginal variance of the log-volatility $\{x_t\}$. It can be proven that the marginal variance of an AR(1) process on the form given in (18) is

$$Var(x_t) = \frac{\sigma^2}{1 - \phi^2} \equiv \eta^2 \tag{39}$$

Thus, several values of η^2 are also tested when evaluating the model performance. Assuming that η^2 is small, it is probably challenging to detect nuances in the parameter values affecting the variance of the log-volatility term.

The parameter values of interest are displayed in table 2. All other parameters will be assumed to be constant for each parameter of interest.

The result of the parameter estimations will be reviewed in the 4.1 and further discussed in 5.1.

Observations (n)	10000	5000	3000	1000	100	15
θ	± 0.1	0				
μ	± 10	± 1	0			
ϕ	± 0.99	± 0.95	± 0.5	0		
σ	0.5	0.1	0.01			
η^2	3	1	0.8	0.1		

Table 2: Parameters of interest, these parameters will be tested. These parameters will be tested to ensure the quality of the model.

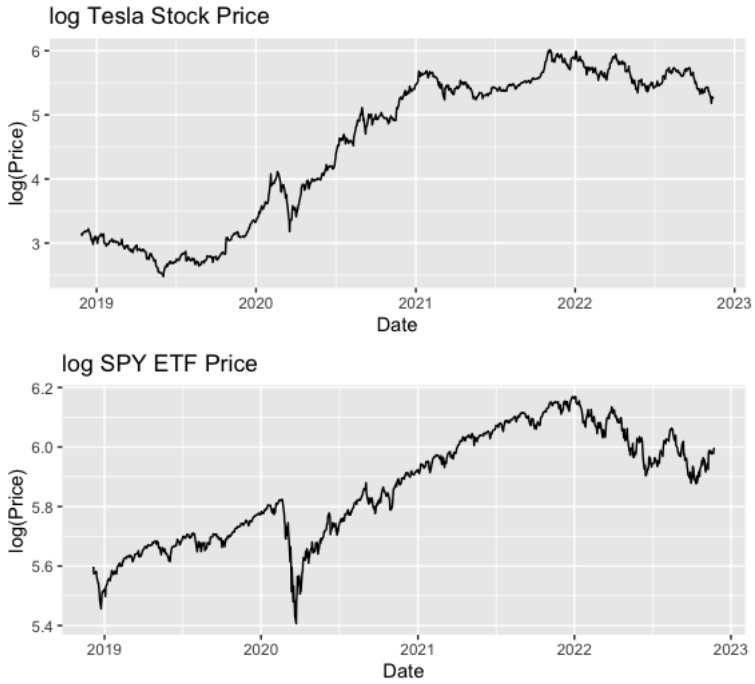

Figure 6: The upper figure is a plot of the logarithmic return of Tesla stock, and the figure below is the logarithmic return of the SPY ETF. Both figures are over the period from 26-November-2018 to 14-November-2022.

3.7 Financial data

In finance, it is considered nearly impossible to predict a stock price based on time series models such as ARMA. Therefore, stocks can are often considered to behave like a random walk. However, predicting a stock's volatility is less difficult. Therefore, when evaluating the performance of SVM, it is natural to test it on financial data.

The difference in performance for GARCH and SVM will be evaluated using two financial time series examples. Each of the examples has different properties that might lead to different results. Since the log-return of a stock (or index) often is considered to follow a random walk, the logarithm of the price will in this paper be treated as the observed values y_t, $t \in \{1, 2, ..., n\}$.

The first time series to evaluate is the log daily closing price of the tesla stock. The time series consists of the 1000 latest daily closing prices. The data set begins on 26 November 2018 to 14 November 2022. The upper plot in figure 6 is a visualization of the logarithm of the stock price.

The last data set consists of the 1000 latest closing prices of the SPY ETF from the same period as the Tesla stock. The SPY ETF is an exchange-traded fund that tracks the S&P 500. The S&P 500 is the index made of the combination of the 500 largest companies traded on the NYSE, NASDAQ, and Cboe. Therefore, the S&P 500 is considered less volatile than other stocks and might be more predictable than the Tesla stock. The logarithmic daily closing price of the SPY ETF is plotted at the bottom of figure 6.

Since Tesla is one of the most valuable companies in the S&P 500, the correlation between the time series is high. This correlation might lead to similar values for the volatility models. Tesla has also had had higher returns and volatility.

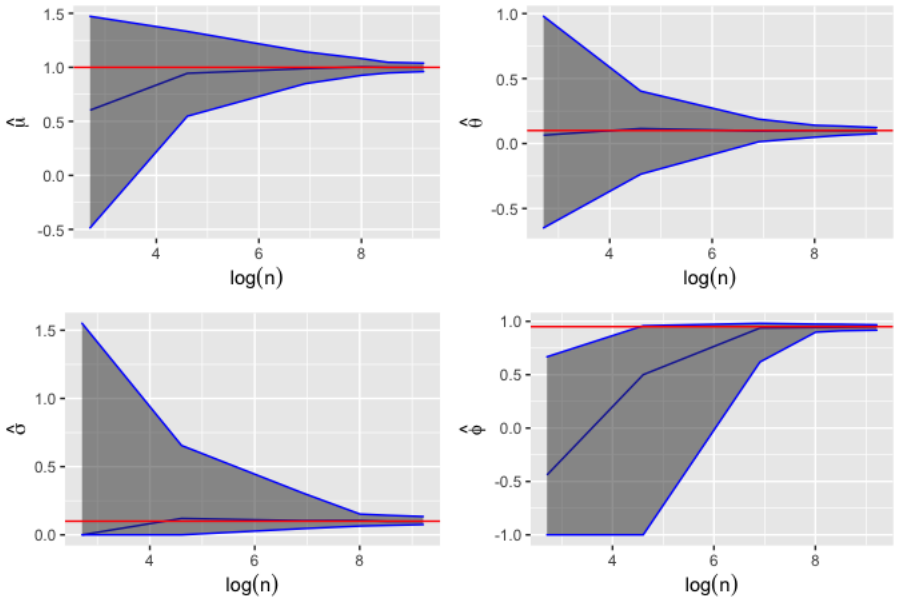

Figure 7: In this figure, the $B = 100$ parametric bootstrap maximum likelihood estimators are estimated for a SV-model for a varying number of observations (n). The true parameters are $\mu = 1$, $\theta = 0.1$, $\sigma = 0.1$, $\phi = 0.95$ and are illustrated by the red lines. The blue lines are the 0.05, 0.5, and 0.95 quantiles of the parametric bootstrap ML estimates.

4 Results

In this section, the results of the parameter estimation of the SVM are presented 5.1. Further, the performance of the SVM is compared relative to a GARCH(1,1) model in 4.2.

4.1 Parameter estimation

In this section, the results of ML parameter estimates are presented. Each actual model parameter is tuned, and its corresponding MLE is evaluated one parameter at a time. Table 2 shows the actual parameters of interest. While one parameter is tuned, the other parameters remain constant. The behavior of all ML estimates will be visualized through plots. For all actual parameter choices, the ML estimates will be calculated $B = 100$ times using parametric bootstrap as described in section 2.8. For each bootstrap sample, the 5%, 50%, and 95% quantiles of the ML estimates are visualized in relation to the actual parameter value.

First, the ML estimates for a different number of observations (n) in the time series are tested. In figure 7, the MLE for each parameter in the SV model is plotted separately as a function of $\log(n)$. Where n is the number of observations. The red lines are the values of the actual parameters used in the data simulation. The values used in this simulation is $\mu = 1$, $\sigma = 0.1$, $\theta = 0.1$ and $\phi = 0.95$. The gray area with the surrounding blue lines is the 90% confidence interval estimated by $B = 100$ parametric bootstrap estimates (2.8). The blue line in the middle is the median of the bootstrapped samples. Note how the MLE converges towards the actual value as n increases.

In figure 8 the values MLE of θ is tuned with the true parameter values $\theta = \{-0.1, 0, 0.1\}$. The other parameter choices are $\mu = 1$, $\sigma = 0.1$, $\phi = 0.95$ and $n = 1000$ observations. The actual value is plotted in red; the blue dot is the median with its corresponding 90% bootstrap confidence interval. Note that for each choice of θ, the confidence interval is relatively big and is approximately the same size. Although $\theta \geq 0$ for most real-world applications, it is interesting to see that the model behaves similarly for all parameter choices.

Figure 9 is plotted in the same way as figure 8, but here the varying for varying μ. $Mu \in$

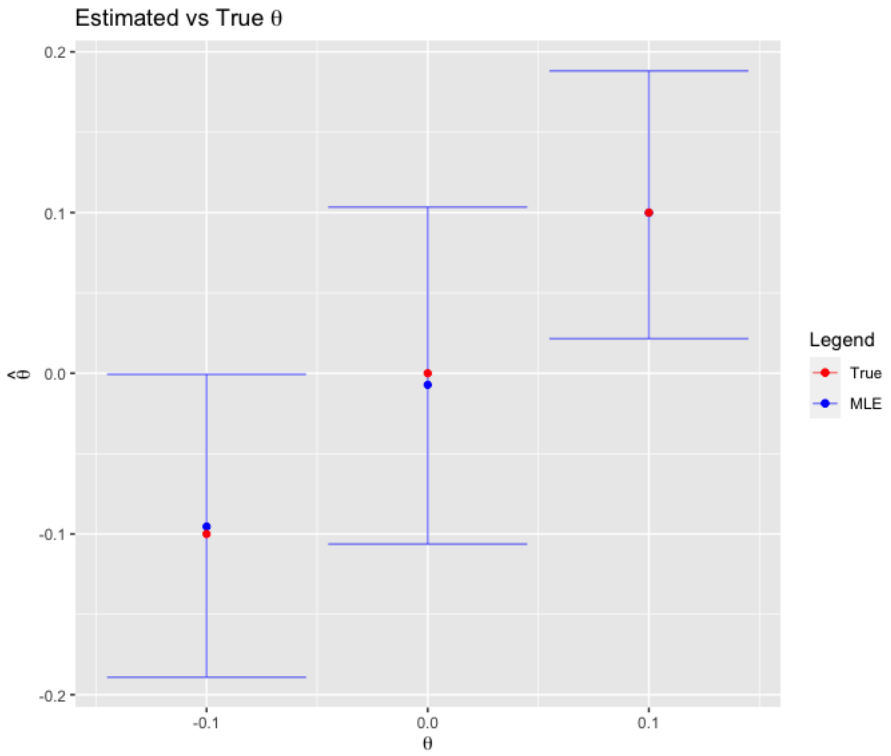

Figure 8: A visualization of the parametric bootstrap ML estimators for different values of θ. For each choice of θ, $B = 100$ parametric bootstrap MLEs are computed, and the 90% confidence interval is displayed in blue as the boundaries of the candlestick. The blue dot is the median MLE, and the red dot is the true value of θ. The other parameters are constant for each simulation and equal to $\mu = 1$, $\sigma = 0.1$, $\phi = 0.95$, $n = 1000$.

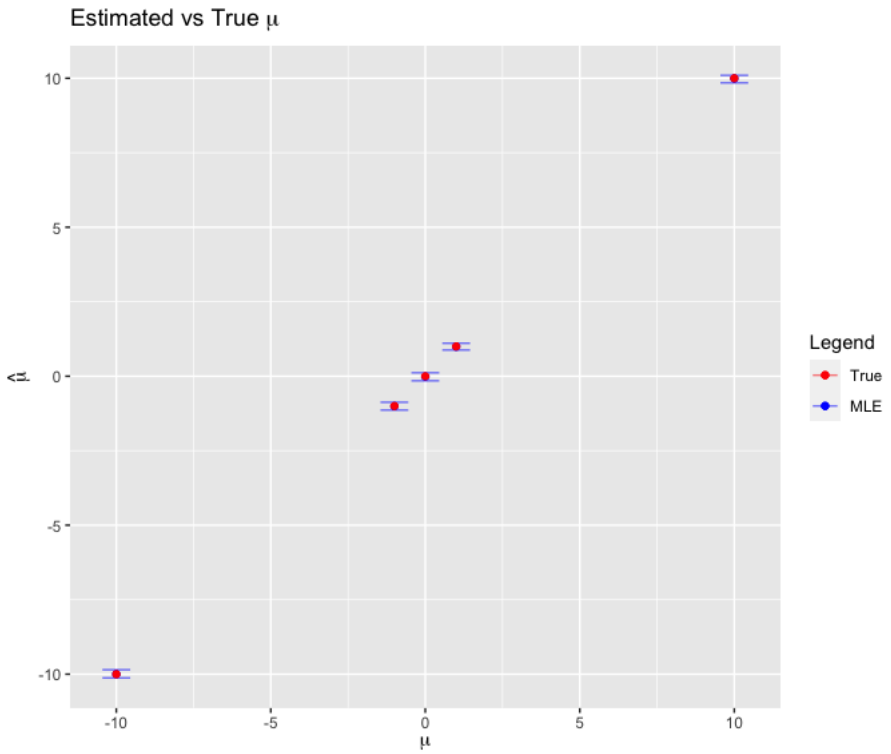

Figure 9: A visualization of the parametric bootstrap ML estimators for different values of μ. For each choice of μ, $B = 100$ parametric bootstrap MLEs are computed, and the 90% confidence interval is displayed in blue as the boundaries of the candlestick. The blue dot is the median MLE, and the red dot is the true value of μ. The other parameters are constant for each simulation and equal to $\theta = 0.1$, $\sigma = 0.1$, $\phi = 0.95$, $n = 1000$.

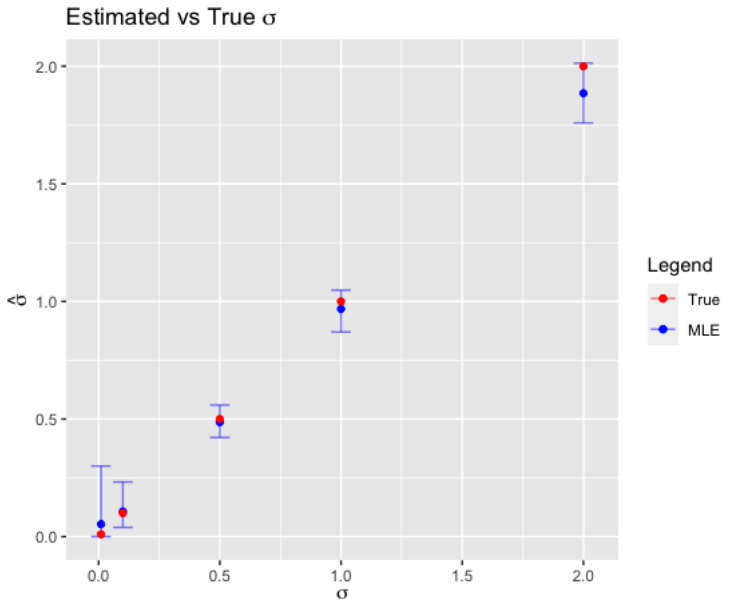

Figure 10: A visualization of the parametric bootstrap ML estimators for different values of σ. For each choice of σ, $B = 100$ parametric bootstrap MLEs are computed, and the 90% confidence interval is displayed in blue as the boundaries of the candlestick. The blue dot is the median MLE, and the red dot is the true value of σ. The other parameters are constant for each simulation and equal to $\theta = 0.1$, $\phi = 0.95$,$\mu = 1$, $n = 1000$.

$\{-10, -1, 0, 1, 10\}$ and $\theta = 0.1$, $\sigma = 0.1$, $\phi = 0.95$, $n = 1000$. It seems like μ is predicted with good accuracy regardless of parameter choice. Similarly to figure 8, the bootstrap confidence intervals are approximately the same size.

In figure 10, the true parameter σ is tuned and displayed, with the bootstrapped 90% interval for each parameter choice. The confidence interval is in blue, with the median as the blue dot. The true value of σ is the red dot. All other variables are constant and set to $\theta = 0.1$, $\phi = 0.95$, $\mu = 1$, $n = 1000$. The confidence interval is much bigger for the smallest and biggest choice of σ ($\sigma = 0.01$, $\sigma = 2$). For these values, the true σ lies on the edge of the confidence interval.

Next, in figure 11, the parameter ϕ is tuned. The MLE of ϕ with its bootstrapped 90% confidence interval is in blue, and the value of ϕ is in red. The other parameters are set to $\theta = 0.1$, $\sigma = 0.1$, $\mu = 1$ and $n = 1000$. Note how the confidence interval for ϕ is approximately between -1 and 1 when the correlation in log-volatility is weak ($\phi \notin \{0.95, 0.99\}$). Also, the ML estimates of the other parameters are still reasonable, indicating that the model only has a problem detecting the dependence in each time step.

From figure 11, it is clear that ϕ is not estimated well. η^2 is defined in (39) and is the marginal variance of the AR(1) part of the volatility. For $\phi = -0.5$ and $\sigma = 0.1$, then $\eta^2 = \frac{0.1^2}{1-(-0.5)^2} = 0.0133$, which is a small variance. Under the hypothesis that the small marginal variance makes it difficult to estimate conditional dependencies in the time series, let $\phi = -0.5$. Figure 12 contains four histograms, where each histogram shows the distribution of the ML estimators of ϕ for different choices of η^2. The red line is the true value of ϕ. The other variables are constant and equal for all histograms ($\theta = 0$, $\mu = 1$, $n = 1000$) In the upper left histogram, η^2 is approximately the same as in figure 11 where $\phi = -0.5$. Note how much better the ML estimates of ϕ are when $\eta^2 = 0.3$.

The presented in this section is discussed in section 5.1.

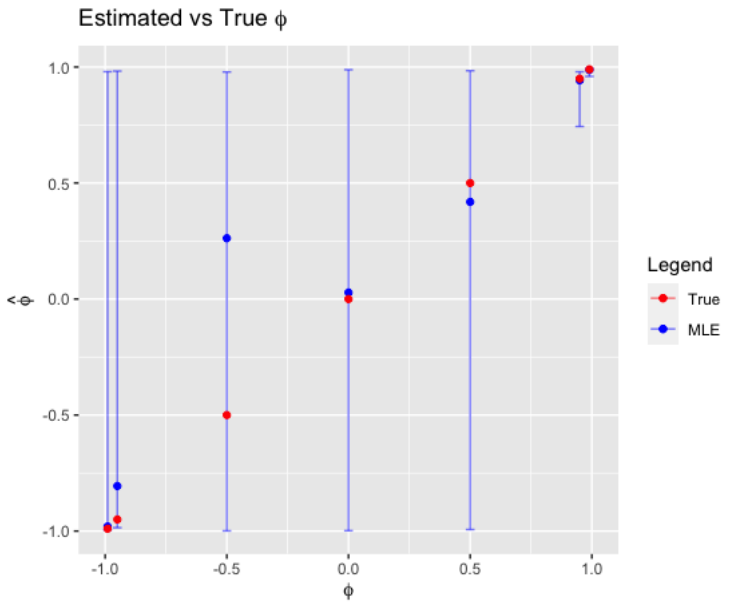

Figure 11: A visualization of the parametric bootstrap ML estimators for different values of ϕ. For each choice of ϕ, $B = 100$ parametric bootstrap MLEs are computed, and the 90% confidence interval is displayed in blue as the boundaries of the candlestick. The blue dot is the median MLE, and the red dot is the true value of ϕ. The other parameters are constant for each simulation and equal to $\theta = 0.1$, $\sigma = 0.1$, $\mu = 1$, $n = 1000$.

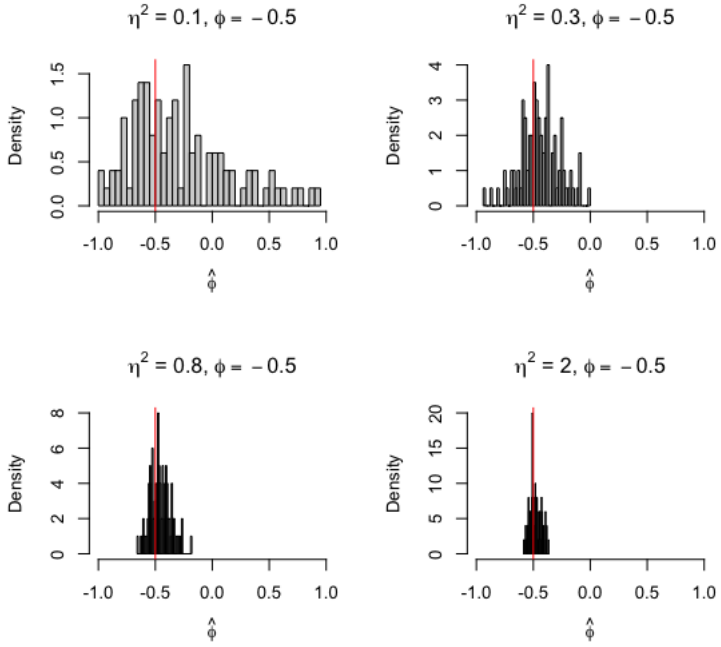

Figure 12: Histogram of the ML estimates for $B = 100$ bootstrap estimates of ϕ. Each model has a different choice of η^2, as displayed above. Other parameter values: $\theta = 0$, $\mu = 1$,$n = 1000$.

,

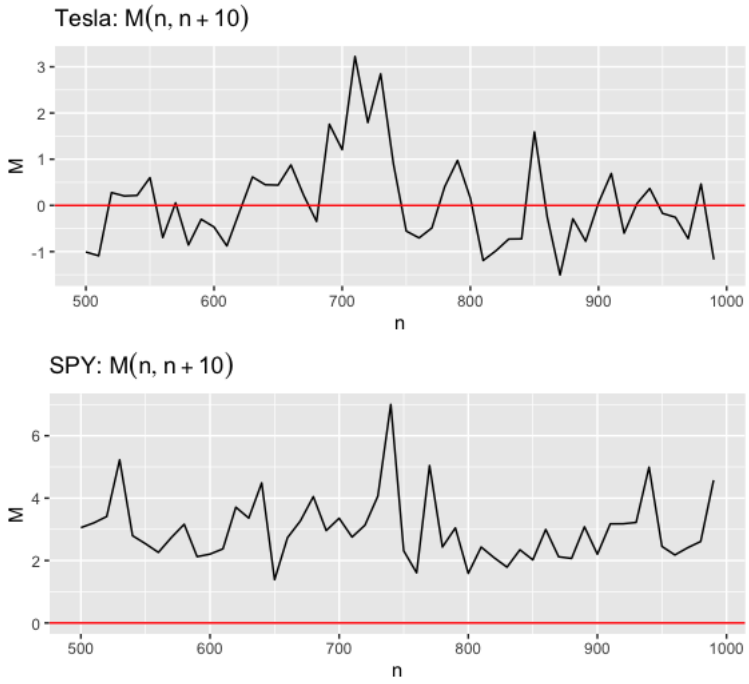

Figure 13: The figure shows how the performance metric M(n,n+10), defined by (34) as a function of the number observations, n. 10 is the number of out of sample observations used for each evaluation. The upper plot is for the Tesla stock, and the lower plot is for the SPY index.

4.2 Cross-validation

This section presents the results from fitting the volatility models to the Tesla and SPY time series. First, each time series is fitted and evaluated several times using the cross-validation described in section 2.7. In this paper, the logarithm of the marginal conditional likelihood, defined in (34), is used for comparing model performance as described in section **??**. The number of observations in the training set is n, and h is the observations in the validation set. In this paper, $h = 10$. The difference between the log-conditional likelihood for SV and GARCH(1,1) model is $M(n, n + 10)$, which is defined by (35). Figure 13 shows how $M(n, n + 10)$ behaves for varying training size $n \in \{500, 510, ..., 990\}$. In the upper figure, the difference in the conditional log-likelihood for the Tesla stock is plotted as a function of n. Likewise, the lower figure plots the conditional difference in the log-likelihood for the SPY ETF. More information on the Tesla and SPY time series is in section 3.7. In figure 13, note how the metric, $M(n, n + 10)$, for the Tesla stock fluctuates between negative and positive values every, while it is always positive for the SPY ETF. For the Tesla stock, this means that neither the SV nor GARCH(1,1) model systematically outperforms the other. Since $M(n, n + 10) > 0$ for all n, there is a systematic difference in model performance, where the SV model fits the time series better.

For each $n \in \{500, ..., 990\}$, the maximum likelihood estimates for the SV and GARCH(1,1) model where calculated (section 3.5). For both time series´ and models, the behavior of the ML estimates is visualized as functions of n. Figure 14 shows the ML estimates of the SV model on the Tesla stock, figure 15 the ML estimates of the GARCH(1,1) model on the Tesla stock, figure 16 the ML estimates of the SV model on the SPY ETF, and figure 17 the ML estimates of the GARCH(1,1) model on the SPY ETF.

If a time series systematically follows the same pattern over time, it would be expected that the ML estimates will converge towards some value as $n \to \infty$. By looking at figure 7, it is clear that as $n \to \infty$ for a model with a systematic pattern, then $\hat{\theta} \to \theta$ for the SV model. Assuming this is the case for any model, including the GARCH(1,1) model. The parameter values in figures 14, 15, 16 and 17 should be converge towards some constant value as n increases. Since this is not the case, it is clear that the behavior of the time series is time-dependent. Further, the cross-validation

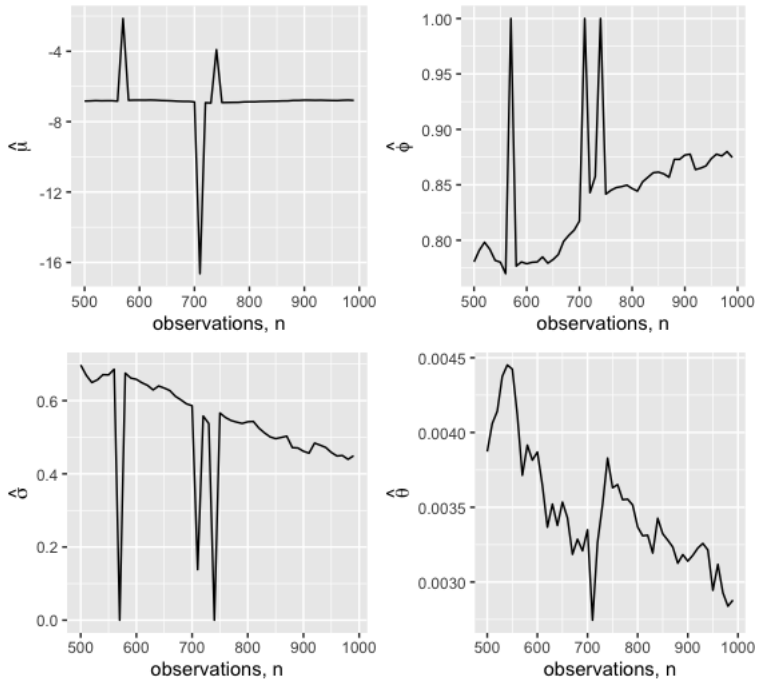

Figure 14: Each figure contains a ML estimate of the SV parameters. These are plotted as a function of n, where n is the number of consecutive log-prices of Tesla stock used in the parameter estimates.

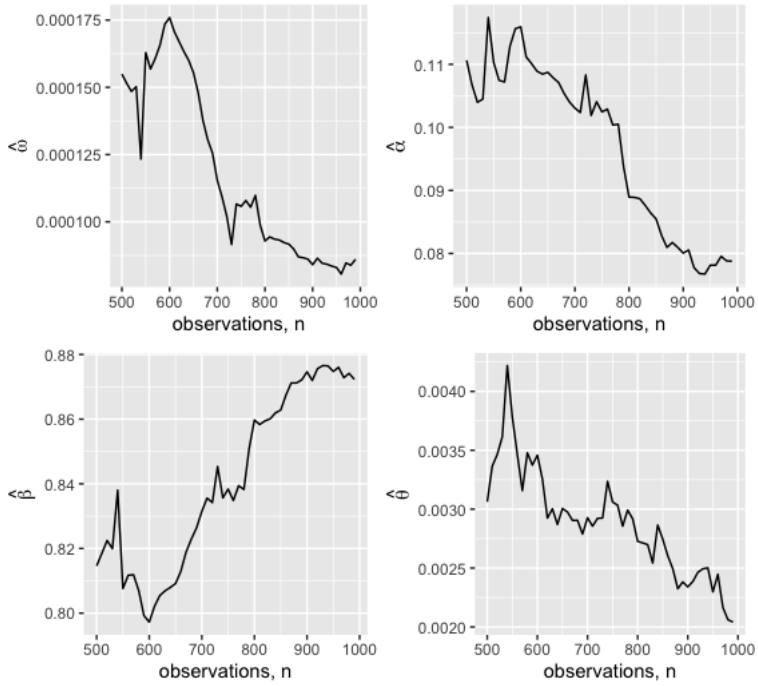

Figure 15: Each figure contains a ML estimate of the GARCH(1,1) parameters. These are plotted as a function of n, where n is the number of consecutive log-prices of Tesla stock used in the parameter estimates.

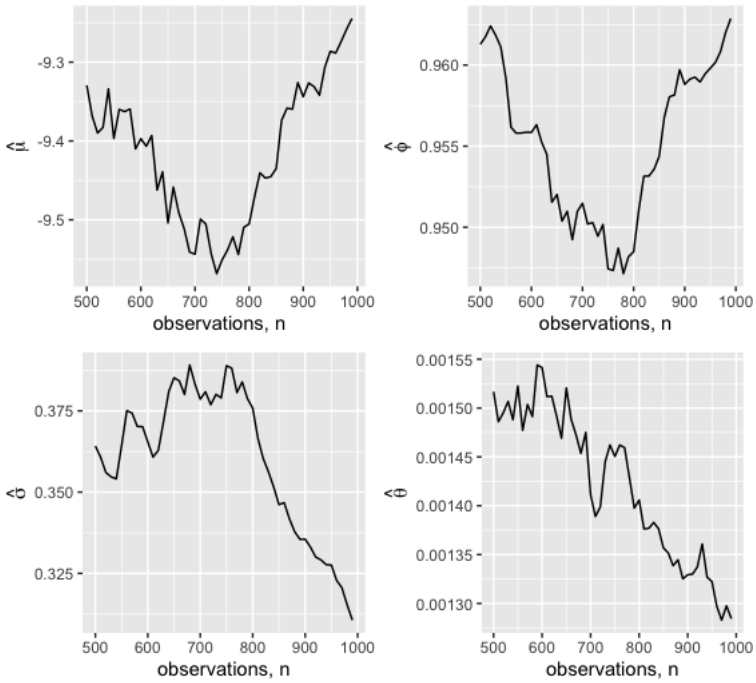

Figure 16: Each figure contains a ML estimate of the SV parameters. These are plotted as a function of n, where n is the number of consecutive log-prices of SPY ETF used in the parameter estimates.

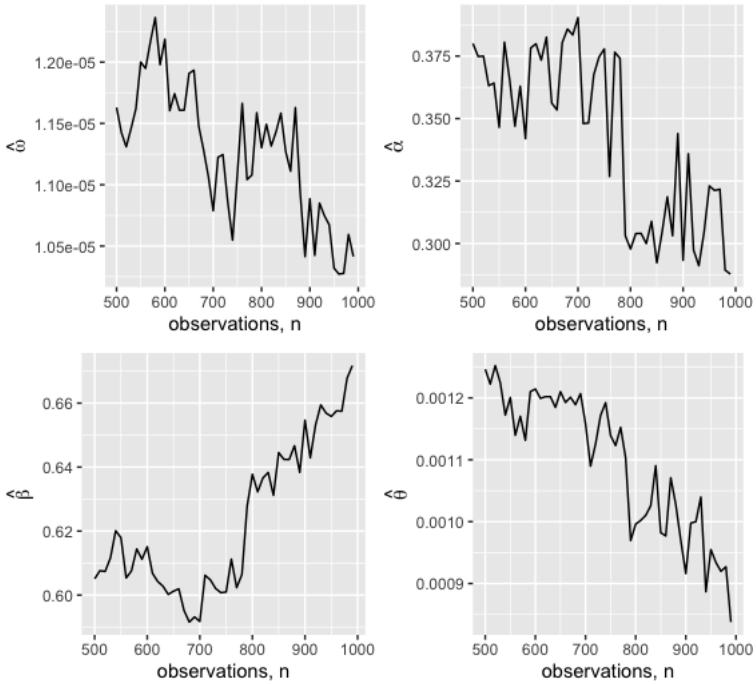

Figure 17: Each figure contains a ML estimate of the GARCH(1,1) parameters. These are plotted as a function of n, where n is the number of consecutive log-prices of SPY ETF used in the parameter estimates.

	L_{SVM}	L_{GARCH}	$SCORE$
Tesla	941	937	4
SPY	1553	1403	150

Table 3: Metrics for the rolling validation of the Tesla stock and SPY ETF. The metrics calculation is explained in section 3.5.

			$\hat{\theta}$	$\hat{\phi}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\omega}$
SVM	Tesla	MLE	0.00273	0.877	−6.77	0.440			
		SD	0.000989	0.0467	0.127	0.0920			
	SPY	MLE	0.00131	0.961	−9.26	0.319			
		SD	0.00024	0.0120	0.258	0.0401			
GARCH	Tesla	MLE	0.00190				0.0776	0.875	0.0000834
		SD	0.00117				0.0162	0.0248	0.0000234
	SPY	MLE	0.000941				0.257	0.656	0.0000124
		SD	0.000280				0.0340	0.0362	0.00000202

Table 4: This table contains the ML estimates and their corresponding standard deviation (SD) for the SV and GARCH(1,1) fitted to the Tesla and SPY time series. How the values are estimated can be read in section 3.3 and 3.1.

described in section 3.5 might be more reliable for training time series with constant length.

The ML estimates for the SVM and GARCH(1,1) for the Tesla stock in figure 14 and figure 15 are volatile. The volatility of the MLEs is more systematic for the GARCH(1,1) model than for the SVM. In the rolling validation of the SVM, then $\hat{\phi} \approx 1$ and $\sigma \approx 0$ for e few instances. Approximately zero variance in the log-volatility model is highly unlikely and will be commented on further in the discussion.

Table 3 presents the final metrics of the rolling cross-validation for Tesla and SPY prices. The table contains the sum of all conditional log-likelihoods (L_{SVM},L_{GARCH}), and difference between those ($SCORE$). Calculations of the metrics are described in section 3.4, by the equations (38) and (36). L_{SVM} and L_{GARCH} are smaller for the Tesla stock than for the SPY ETF. The big difference in the L_m for the SPY ETF and the Tesla stock indicates that the SPY is less volatile and more predictable than Tesla.

$SCORE$ is a metric for comparing the SVM and GARCH(1,1) models. $SCORE > 0$ is an indication that SVM outperforms GARCH(1,1) on the given time series. Similarly, $SCORE < 0$ indicates that GARCH(1,1) outperforms SVM. From table 3, it is clear that SVM outperforms GARCH(1,1) for both the Tesla stock and the SPY ETF. The $SCORE$ for SPY is much greater than zero, indicating that the SV model is a better fit for this time series.

Table 4 shows the ML estimates of each parameter for each model and time series with its corresponding standard deviation (SD). The MLEs are obtained by fitting the SVM and GARCH(1,1) to the full Tesla and SPY time series. The standard deviations (SD) are approximated by using the δ-method as defined in equation (15) for the SV model. Similarly, the standard deviations of the GARCH(1,1) models are approximated using the delta method, but by using a numerical approximation of the Hessian, gotten from the **optim**-function in R.

In table 4, $\hat{\theta}$ can be considered the daily expected increase of the log price of an asset. For both time series,´ the difference in the ML estimate of θ is big. For the SVM $\theta = 0.00273$ and for GARCH(1,1) $\theta = 0.00190$, the expected daily increase of the log-price is significantly higher for the SVM than for the GARCH(1,1) model. Since $\hat{\phi}$ is smaller for Tesla than for SPY, according to the SVM, the day-to-day volatility is higher for SPY than for Tesla. For the GARCH(1,1) model, $\hat{\alpha}$ estimates the effect of a sudden burst in volatility on tomorrow's volatility. Therefore, a sudden burst in volatility impacts the volatility of SPY more than Tesla. Further, $\hat{\beta}$ indicates the day-to-day dependence between in volatility, which in this case is lower for SPY than for Tesla. This is a contradiction to the SVM model.

	$\hat{V}_t^{SVM}$	$\hat{V}_t^{GARCH}$
Tesla	0.00174	0.00177
SPY	0.000185	0.000434

Table 5: Comparing the expected marginal variance of SVM and GARCH(1,1), defined in (30) and (31), for the Tesla and SPY examples.

In table 5, the expected marginal variance for all fitted models is computed. The computation of the marginal variance is elaborated in section 3.4. Assuming that both models fit the data equally well and that $\hat{\theta}$ is approximately the same, it should be expected that the marginal variance of the SV and GARCH(1,1) is the same. The marginal variances for both models fitted to the Tesla time series are approximately the same. However, they are significantly different for the SPY index.

5 Discussion

In this section, all results are discussed. In subsection 5.1, the results obtained in subsection 4.1 are discussed. I am following by subsection 5.2, where the results from 4.2 are discussed.

5.1 Parameter estimation

In section 4.1, the confidence interval for different ML estimates is visualized through different plots. The default parameter choice for the SV model was $\theta = 0.1$, $\sigma = 0.1$, $\phi = 0.95$, $\mu = 1$.

The default case's behavior was visualized in figure 7. Here the number of observations was chosen as a variable. It is evident that for a high number of observations n, the ML estimates converge toward the actual parameter values. This indicates that the model works as expected and could be reliable when applied to real-world applications, as long as the time series of interest has sufficiently many observations.

However, this might only be true for some models. In figure 7, the actual value of ϕ is set to 0.95, which can be considered a high value as $|\phi| < 1$. It can be safe to assume that it is easier to recognize clear patterns in volatility for high values of ϕ than for smaller values of ϕ. Since the log-volatility follows an AR(1) process, a small value of ϕ indicates a rapid change in conditional volatility for each time step. From figure 11, it is clear that the ability of the model to estimate the true ϕ is poor when $n = 1000$ and $\sigma = 0.1$.

The 90% confidence intervals of $\hat{\phi}$ in figure 11 are fitted using bootstrap, as previously explained. For every parameter value tested, $B = 100$ ML estimators are calculated for ϕ. When the actual value of ϕ is not 0.99 or 0.95, the confidence interval for $\hat{\phi}$ is approximately between -1 and 1. This means that more than 5% of the ML estimators are approximately -1 ($\hat{\phi} \approx -1$) and more than 5% of the ML estimators are approximately 1 ($\hat{\phi} \approx 1$). For $\phi = -0.5$ and $\sigma = 0.1$, the probability that $\phi \approx 1$ or $\phi \approx -1$ is greater than 10%. Since $-1 < \phi < 1$, and the confidence interval is the same size, the results are not satisfying.

From figure 8 in the result section, it is clear that the confidence interval is quite large here. For all choices of θ, the confidence interval is approximately 0.2. This is for simulations with 1000 observations. For instance, when looking at financial data, it would never be the case that the average daily return is 0.1. This is something to be aware of when the model is applied to the example time series. In figure 7, it is clear that as observations increase, the confidence interval of θ decreases, which is good.

To put the importance of estimating the correct model in perspective, the financial market has 255 days of trading each year. Therefore, it takes approximately four years to generate 1000 daily closing prices (data points). If **TMB** cannot be used to fit a decent model with four years' worth of data, that is a critical flaw in a financial model. Financial time series have constantly varying volatility with positive correlation. The SV model is reliable for strong positive correlations. So, the SV model implemented in **TMB** can describe a financial time series well.

To conclude, if the marginal variance of the log-volatility is high, then the model has a good chance of being reliable. The marginal variance of the log-volatility impacts the number of observations needed to get a reliable model.

The results from fitting a SV and GARCH(1,1) of the Tesla stock and SPY ETF are discussed in the next section.

5.2 Model Comparison

In this section, the results from section 4.2 are discussed. Starting with the metrics from the cross-validation, then looking at the ML estimates. Then, finally, discussing the model parameters fitted to the whole time series.

As described in section 4.2, in figure 13, the difference between the conditional log-likelihood of the models ($M(n, n+10)$) are plotted as functions of $n \in \{500, ..., 990\}$. Every time M is below the red line, the GARCH(1,1) model outperforms the SVM for the validation set $\{n, ..., n+10\}$. For the Tesla time series, this happens regularly throughout the cross-validation. This indicates that there are times when the GARCH(1,1) model is better at predictions than the SVM. In table 3, $SCORE = 4$ for the Tesla series, indicating that the SVM slightly outperforms the GARCH(1,1) model in the long run. Because the value is so close to zero, it is difficult to determine if the positive $SCORE$ is random.

For SPY, $M(n, n+10) > 0$ for all n. This indicates that the fitted SVM is much better at predicting the behavior of the SPY ETF for all validation sets. This can also be seen by looking at table 3, where $SCORE = 150$ for the SPY ETF. This is a much bigger $SCORE$ than for the Tesla models.

When comparing L_{SVM} and L_{GARCH} for the SPY and Tesla time series in 3, clearly L_m is much higher for SPY than for Tesla. The big difference is that SPY is less volatile than Tesla stock. For a less volatile time series, it is easier to predict the next value. Thus, L_m is generally greater for the SPY models. A similar argumentation can be used when explaining that $L_{SVM} \gg L_{GARCH}$ for the SPY models. For SPY, the GARCH(1,1) model overestimates the marginal volatility of the model relative to the SVM.

In table 5, the marginal variances are compared for the whole time series. For the Tesla models, the marginal volatility is approximately equal, but for the SPY models, the marginal variance of the GARCH(1,1) model is much larger than for the SVM. Even though table 5 on the whole time series, it is a good indication of the value of the marginal volatility for the training time series used in the validation models.

Assume that the marginal volatility has approximately the same behavior as in table 5 for each validation set in the cross-validation. For SPY, this would explain why the $M(n, n+10) > 0$ for all validation sets. Similar behavior for all validation sets would mean that the marginal volatility is approximately equal for the Tesla SV and GARCH(1,1) models. If the marginal variances are similar, then it would make sense that the out-of-sample validation in the upper figure 13 behaves randomly as well. Both hypotheses are in line with the results. However, there are some variations in the ML estimates throughout the validation set.

If a time series systematically follows the same pattern over time, it is expected that the ML parameter estimates will converge towards some true parameter values as $n \to \infty$. By looking at figure 7, it is clear that as $n \to \infty$ for a model with a systematic pattern, then $\hat{\theta} \to \theta$ for the SV model. Assuming this is the case for any model, including the GARCH(1,1) model. The parameter values in figures 14, 15, 16 and 17 should converge as n increases. Since this is not the case, it is clear that the behavior of the time series is time-dependent, which means that the behavior of the time series changes over time. If the properties of a time series change over time, this contradicts the approximately constant marginal variance assumption. Further indicating that the cross-validation split described in section 3.5 might be better for both models if the observations used in the training set are constant instead of increasing.

SPY is, by construction, less volatile than the Tesla stock. Therefore, the parameter estimates are also expected to be less volatile. The first 500 observations are used for parameter estimation in the first training time series. Each observation is equally weighted. Therefore, the observation on the first day is equally important as the observation on the last day. Further, this means that the first five hundred observations account for over half the weight for all validation sets (half for the final models). In other words, the behavior of the first 500 observations plays a vital role in the ML estimates for the parameters for all models. Assuming that the GARCH(1,1) SPY model is a horrible fit for the first 500 days and that the SV SPY model is a much better fit. This can explain the superior behavior of the SVM for all validation sets.

In figure 14, it can be seen that $\hat{\phi} \to 1$ and $\sigma \to 0$ for some validation sets. This indicates that the model fails to converge in these validation sets. In section 4.1, it became clear that the **TMB** implementation (3.2) is not always reliable. However, the reliability of the model increases as the log marginal volatility (η^2) increases. Looking at table 4 for the full SV model for the Tesla stock, the log marginal volatility can be calculated as $\hat{\eta}^2 = 0.839$. In figure 12, it was

seen that for $\hat{\eta}^2 \approx 0.8$, the ML estimates are close to the actual parameter. Regardless of the parameter estimates in figure 14, it can be seen in figure 13 that $M(n, n+10)$ behave similarly for all values. The strange parameter estimates seem to have little impact on the conditional marginal log-likelihood. Therefore, the outliers can be overlooked. However, pay attention to the final model parameters and note if $\sigma \to$ and $\phi \to 1$.

When looking at the GARCH(1,1) model defined by (27) and the SVM defined by (18), both models have the same number of parameters. This should mean that both models have a similar ability to predict volatility. However, SVM has two random terms, while GARCH(1,1) only has one. Therefore, the SV model should be more flexible than the GARCH(1,1).

A final remark on the performance of the GARCH(1,1) model relative to the SVM. The author of this paper has implemented both models. Therefore, there might be some errors in the code affecting the results.

6 Summary

In this paper, the R package **TMB** has been used to perform inference on a SV model with a logarithmic AR(1) volatility term. **TMB** was used to compute ML estimate of the model parameters by computing an approximation of the marginal maximum likelihood and maximizing with respect to the parameters. When evaluating the model's performance, it was found that two factors mainly affected the parameter estimates' accuracy. It was shown that an increase in observations in the time series and/or an increase in the marginal variance of the AR(1) term led to more reliable parameter estimates. Finally, it was seen that for low values of the AR-coefficient, ϕ, the accuracy of the ML estimates dropped.

Further, the performance of the SVM was tested relative to a GARCH(1,1) model. The difference in model performance was evaluated on the logarithm of the daily closing prices of the Tesla stock and SPY ETF. The SVM outperformed the GARCH(1,1) model for both time series, indicating that the given stochastic volatility model is the better model. For the Tesla example, the marginal volatility of the SV model was slightly lower than the marginal volatility of the GARCH(1,1) model, and the *SCORE*-metric was somewhat in favor of the SVM. For the SPY example, the marginal volatility of the SV model was much lower than for the GARCH(1,1) model, and the *SCORE* was significantly in favor of the SVM. Therefore, there is a correlation between model marginal volatility and model performance.

In Martino et al., 2011, it was stated that the additional error term in the SV model makes it more flexible than a GARCH model. This assumption was not contradicted in this paper.

The next step could be to explore the correlation between marginal volatility and model performance by looking at more time series examples. Otherwise, one could look at more advanced methods like SV models with log-AR(p) term.

References

Baydin, Atilim Gunes, Barak A. Pearlmutter and Alexey Andreyevich Radul (2015). 'Automatic differentiation in machine learning: a survey'. In: *CoRR* abs/1502.05767. arXiv: 1502.05767. URL: http://arxiv.org/abs/1502.05767.

Bergmeir, Christoph and José M. Benıtez (2012). 'On the use of cross-validation for time series predictor evaluation'. In: *Information Sciences* 191. Data Mining for Software Trustworthiness, pp. 192–213. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2011.12.028. URL: https://www.sciencedirect.com/science/article/pii/S0020025511006773.

Bishop, Christopher M. (1998). 'Latent Variable Models'. In: *Learning in Graphical Models*. Ed. by Michael I. Jordan. Dordrecht: Springer Netherlands, pp. 371–403. ISBN: 978-94-011-5014-9. DOI: 10.1007/978-94-011-5014-9_13. URL: https://doi.org/10.1007/978-94-011-5014-9_13.

Bollerslev, Tim (1986). 'Generalized autoregressive conditional heteroskedasticity'. In: *Journal of Econometrics* 31.3, pp. 307–327. ISSN: 0304-4076. DOI: https://doi.org/10.1016/0304-4076(86)90063-1. URL: https://www.sciencedirect.com/science/article/pii/0304407686900631.

Brockwell, Peter J. and Richard A. Davis (2016a). 'ARMA Models'. In: *Introduction to Time Series and Forecasting*. Cham: Springer International Publishing, pp. 73–96. ISBN: 978-3-319-29854-2. DOI: 10.1007/978-3-319-29854-2_3. URL: https://doi.org/10.1007/978-3-319-29854-2_3.

— (2016b). 'Time Series Models for Financial Data'. In: *Introduction to Time Series and Forecasting*. Cham: Springer International Publishing, pp. 195–226. ISBN: 978-3-319-29854-2. DOI: 10.1007/978-3-319-29854-2_7. URL: https://doi.org/10.1007/978-3-319-29854-2_7.

Kristensen, Kasper et al. (2016). 'TMB: Automatic Differentiation and Laplace Approximation'. In: *Journal of Statistical Software* 70.5, pp. 1–21. DOI: 10.18637/jss.v070.i05.

Martino, Sara et al. (2011). 'Estimating stochastic volatility models using integrated nested Laplace approximations'. In: *The European Journal of Finance* 17.7, pp. 487–503. DOI: 10.1080/1351847X.2010.495475. eprint: https://doi.org/10.1080/1351847X.2010.495475. URL: https://doi.org/10.1080/1351847X.2010.495475.

Pedregosa, F. et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Williams, Brandon (2011). 'Betreuung: Prof. Dr. Rainer Dahlhaus'. In.

7 Appendix

All R code can be found on my github:

Link Here