Shrinkage estimation in ARMA-GARCH regression models with an application in Bitcoin returns

by

Zola Mary-Jean Sibanda

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae in Mathematical Statistics

In the Department of Statistics
In the Faculty of Natural & Agricultural Sciences
University of Pretoria
Pretoria

December 2019

I, *Zola Mary-Jean Sibanda,* declare that this mini-dissertation (100 credits), which I hereby submit for the degree Magister Scientiae in Mathematical Statistics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

____

Signature - Zola Mary-Jean Sibanda

____

Date

Contents

List of Figures

List of Tables

Summary

We focus on the extensions of autoregressive conditional heteroscedastic (ARCH) models and the generalised autoregressive conditional heteroscedastic (GARCH) models applied to financial data. Volatility is observed in financial time series as a response to information or news, which in most cases is unknown beforehand. Although, in certain situations, the timing of information provided may not be a surprise (e.g. announcements of mergers or initial public offerings (IPOs), etc.), giving rise to some aspects of volatility being predictable. Even though volatility is a latent measure in that it is not directly observable but given ample information, it can be estimated. With the uncertainty of risk on financial assets, it would be an inadequate assumption that a constant variance exists over a given time period which is assumed when using ordinary least squares estimation. In the past, linear regression models were used to predict relationships between macro-economic variables but when heteroscedasticity is present, one might still obtain unbiased regression parameter estimates with too low standard errors, which will influence the true sense of precision. The ARMA-GARCH regression model is one of many extensions of the GARCH process with respect to the conditional mean. This dynamic model allows for both the conditional mean and conditional variance to be modelled by the ARMA process and the GARCH process respectively. More specifically, in this mini-dissertation, we develop shrinkage estimation techniques for the parameter vector of the linear regression model with ARMA-GARCH errors. For the purpose of shrinkage estimation, we will be assuming that some linear restriction hold on the regression parameter space. From a practical point of view, specifying a set of logical restrictions plays an important role in economic and financial modelling. We conducted an extensive Monte Carlo simulation study to assess the relative performance of the proposed estimation techniques compared to the existing likelihood-based estimators. The application of our research is considered in the estimation and modelling of Bitcoin returns and testing the significance of the interest in the topic of cryptocurrencies as well as the impact of which traditional financial markets may have on Bitcoin and the cryptocurrency market.

Keywords: ARMA-GARCH regression, Bitcoin return, maximum likelihood estimation, Preliminary test estimator, Shrinkage estimator.

Acknowledgements

I would like to thank the following, without whom I would not have been able to complete this mini-dissertation, and without whom I would not have made it through my masters degree. Thank you to my supervisors, Dr. Kleyn and Prof. Arashi, for providing guidance and feedback throughout this project. I would like to say a special thank you to Dr. Kleyn for the two years we have worked together on completing this research. You encouraged me, gave me ample support and allowed me to make this research my own. I would also like to thank Prof. Arashi, especially, whose insight and knowledge into the subject matter steered me through this research. A special thank you to Dr. Norouzirad, for all the assistance you provided concerning the coding. I was able to complete the analysis with your suggestions and I appreciated the timely feedback you gave me. And my biggest thanks to my parents and my son, Owethu, for all the support you have shown me throughout my studies. I can never thank you enough for all you have done for me.

The financial assistance of the Center of Artificial Intelligence Research (CAIR) towards this research is hereby acknowledged.

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Number 71199)

1 Introduction

With the rapid development of technology in the last few decades since the Dot-com bubble burst, online transactions are the new norm; from consumers making more purchases online (i.e. the birth of e-commerce era) and a greater number of individuals banking online for its efficiency and simplicity. The investment sector has felt the spill over of e-commerce as well. It has also been reported that with the fourth industrial revolution beckoning, that we have also generated about 90% of the world's entire data between the years of 2015 and 2016 which only continues to grow at an increasing rate. With all this information available, it is crucial to make sense of the data in order to make well-informed decisions. Technological advancements have therefore allowed us to better collect, store, analyse and share large data sets with greater ease across different pools, within businesses and across countries with little to no cost. As a result, this allows information to be virtually accessible to anyone looking for it. This has also brought about great concerns over the aspect of data privacy and more individuals preferring online contact that provides some anonymity given that some sort of digital trail is left behind from all online activities.

The global financial crisis between 2007 and 2009 led to a restructuring of the traditional financial institutions and a transformation of financial markets known today. As a result, more unorthodox measures and solutions have been increasingly applied out of necessity as a safeguard, to prevent another global financial crisis from taking place again and to combat the issues around privacy and security. This eventually led to the introduction and evolution of digitised currencies. The year 2009 marked the inception of first digital currency or cryptocurrency known as the Bitcoin (BTC) designed to take advantage of this internet age. The popularity and interest of cryptocurrencies have increased exponentially in the last decade with well over a thousand cryptocurrencies available in the market. They are also known as AltCoins because they closely resemble Bitcoin technology by adopting its main features and differing in only some. Bitcoin is the first cryptocurrency that holds the largest cryptocurrency market capitalisation of 109 billion USD as of July 2018. Bitcoin is also increasingly being integrated as a medium of exchange for daily operations. Its price is highly speculative; greatly determined by people's interest in the asset which makes it highly volatile and therefore a risky investment. The various news reports around Bitcoin such as defaults, attacks, and government regulation have caused both extremely positive and negative effects on the Bitcoin price. These factors also influence the Bitcoin's volatility. Bitcoin's volatile nature has ensued a lot of curiosity and now there is considerable participation in the study and analysis of Bitcoin returns from traditional financial institutions as well as in academia.

The focus in this mini-dissertation is to show that despite Bitcoin's high volatility, it can still be modelled and predicted using statistical knowledge of financial time series. The goal is to provide greater insights into the behaviour of Bitcoin, particularly focusing on its volatility. We aim to provide a deeper understanding of digitised assets and extend on the current literature available in Bitcoin volatility and forecasting while applying GARCH modelling techniques.

1.1 Financial Time Series Analysis

Financial time series provide information about the valuation of assets or other commodities and their developments over time. The price data, as well as all transactions, are collected and recorded on a financial market at high frequencies (e.g. daily, hourly etc.) and monitored on a continuous basis. The analysis of these financial time series have been studied for decades; the theory was developed in the stochastic processes space and one of the earliest applications was of the autoregressive (AR) model by Yule [64] in 1927 and later by Walker [58] in 1931. AR models base their future predictions on past information such that the variable is linearly dependent on at least one of its lags (i.e. previous terms), producing a type of difference equation. Prior to the development of AR models, the moving averages (MA) model was used. The family of MA processes are a type of smoothing technique for time series data points; the error terms are modelled such that they form a linear combination of the past innovations that occur concurrently and at various past times. It was only until the 1970s that all types of time series modelling procedure developed at that time were published by Box and Jenkins [11]. It was also these two authors who popularised the Box-Jenkins model also referred to as the autoregressive moving average (ARMA) model, initially described in Whittle's [60] 1951 thesis. The parsimonious ARMA model is a combination of both the AR model and the MA model. Box and Jenkins [11] later generalised the ARMA model to include a differencing operator raised to an integer power called the autoregressive integrated moving average (ARIMA) model. Many of these models described are based on the mean of a variable and its time-varying attributes for the purpose of forecasting, but no methods were developed for the variance at the time.

Financial time series of price data are typically nonstationary which makes them unpredictable and spurious resulting in some difficulty in building models for forecasting as parameters of interest may no longer be constant. Therefore, analyses are usually performed on returns data which are generally more stable since they have statistical properties that can be estimated. A financial time series also exhibits some other characteristics which include time-varying variability referred to as heteroscedasticity; volatility clustering which was first realised by Mandelbrot [36] who confirmed that future variation was conditional on past variation which meant that large (small) changes in volatility which would, in turn, tend to lead to large (small) changes in future volatility of either sign. Mandelbrot's [36] concept of volatility clustering was built on Bachelier's [3] work on the financial applications of random-walk models dating back to the early 1900s.

From these characteristics, volatility is of importance in the analysis of financial time series because of the significant trade-off between risk and return. Volatility can be described as a measure of risk due to the size of a return's modelling error or as defined by Tsay [54] as the underlying asset return's conditional standard deviation. Volatility has uses in possibly determining one's financial risk position, or asset allocation and or other forms of time series analysis. Volatility is a response to information or news which in most cases is unknown beforehand. Although, in certain situations, the timing of information provided may not be a surprise (e.g. announcements of mergers or initial public offerings (IPOs) etc.) giving rise to some aspects of volatility being predictable. Even though volatility is a latent measure in that it is directly unobservable but given ample information, it can be estimated. With the uncertainty of risk on financial assets, it would be inadequate to assume that there exists a constant variance over a given time period as with ordinary least squares (OLS) estimation. This obviously contradicts reality and thus the autoregressive conditional heteroscedasticity (ARCH) model was first introduced by Engle [17] in 1982 which revolutionised the way in which we model volatility. This dynamic model of volatility takes weighted averages of the past squared forecast errors. Historical data can be used to estimate the weights such that more weight is given to the most recent information and less on the distant information. As profound as the ARCH models may be, it assumes that the environment is stable and thus has the inability to capture incidences of irregularity (e.g. crashes in the market, mergers and acquisitions or other news effects etc).

Prior to the inception of ARCH models, there was no formal way of modelling volatility especially in the context of financial data that took into account some of the important characteristics mentioned above. For example, the concept of *implied volatility* is derived under certain assumptions and then used in determining option prices under the Black-Scholes model but tends to overestimate volatility; Mandelbrot [36] also used a recursive formula in order to estimate the variance at a given time; alternatively, Klein [32] would take a 5-period moving average of the variance as estimates of a 10-period moving average sample mean. ARCH models, therefore, allow us to make more sound financial forecasts and predictions. Since then, there has been vast amounts of literature that extend upon Engle [17] ideology such as Bollerslev [8] who proposed a generalisation of the ARCH model defined as the generalised autoregressive conditional heteroscedastic (GARCH) model. The GARCH model is widely used because it generalises the autoregressive ARCH into an autoregressive moving average (ARMA) model with the assumption that positive and negative error terms have a symmetric effect on volatility referred to as the leverage-effect. The GARCH volatility measure is such that the future volatility is based on the past information of its squared returns and conditional on its past volatility. Both the ARCH and GARCH models are widespread variance models built to treat heteroscedasticity and therefore particularly used in dealing with financial time series. The ARCH-class family has many extensions due to its success in financial applications to satisfy more real-world scenarios. In 1987, Engle et al. [16] proposed an extension of the ARCH model called the ARCH-M model which is based on the concept that the conditional mean of a financial asset would be influenced by its conditional variance which in turn affects the return on the asset. This was then applied to both short- and long-term interest rates which revealed that short rates had a stronger time-varying risk premium than the long term rates. In 1991, the exponential GARCH (EGARCH) model was proposed by Nelson [38] because he recognised that there were significant differences in the way negative returns affected the volatility predictors compared to positive returns (i.e. volatility reacted asymmetrically to positive or negative news). Nelson's [38] contribution has made a massive impact on the development of volatility measures today. Then between 1991 and 1993, Zakoian [65] published a new modification of the ARCH model called the Threshold Heteroscedastic (TARCH) models which was then generalised by Rabemananjara and Zakoian [42] named the Threshold GARCH (TGARCH) model. The TARCH was designed such that the distribution of innovations are divided into disjoint intervals which are then approximated by a piecewise linear function for the conditional standard deviation and conditional variance; by Zakoian [65] and Glosten et al. [23] respectively. The TGARCH model separates the impact of past shocks by using zeros as thresholds. Thereafter, Bera and Higgins' [7] paper discussed the properties of many of the ARCH-class family models that had been developed by 1993 and also looked at the testing and estimation of these models. The family of ARCH-class models have also been extended to consider multivariate occurrences. In 2015, Messaoud and Aloui [37] made use of copula functions to examine the dependence of asymmetries in order to calculate some risk measure. The Glosten-Jagannathan-Runkle generalised autoregressive conditional heteroscedastic (GJR-GARCH) model was the main focus of the study with its parameters estimated using the generalised Pareto distribution.

As has been previously mentioned, financial time series analysis has received considerable attention and explosively developed with the increase of data available and the vast amounts of literature produced. The main objective of financial time series analysis is to help us gain a better understanding of how financial data behave. We particularly look at the variance of a time series as it is of key importance since the future price of an asset is unknown and uncertain. This can be described by a probability distribution; therefore, statistical methods are used to investigate and model financial assets. Financial models describe in detail the movements of an asset or commodity over time by replicating the time series, as a result, can be used to simulate and determine their future behaviour in order to make the most optimal decisions; this is known as risk management.

The following figures give over 700 observations of the Bitcoin price and log returns respectively, taken at daily intervals between the period of March 31st, 2016 to March 31st, 2018.

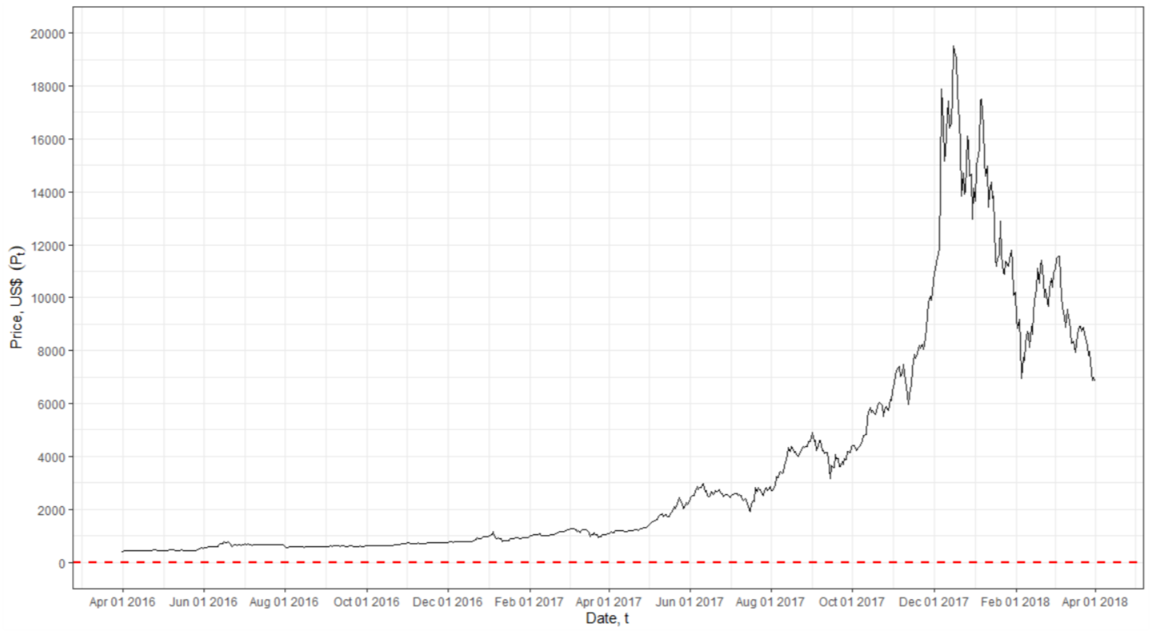


Figure 1: Time plot of USD/BTC daily rate.

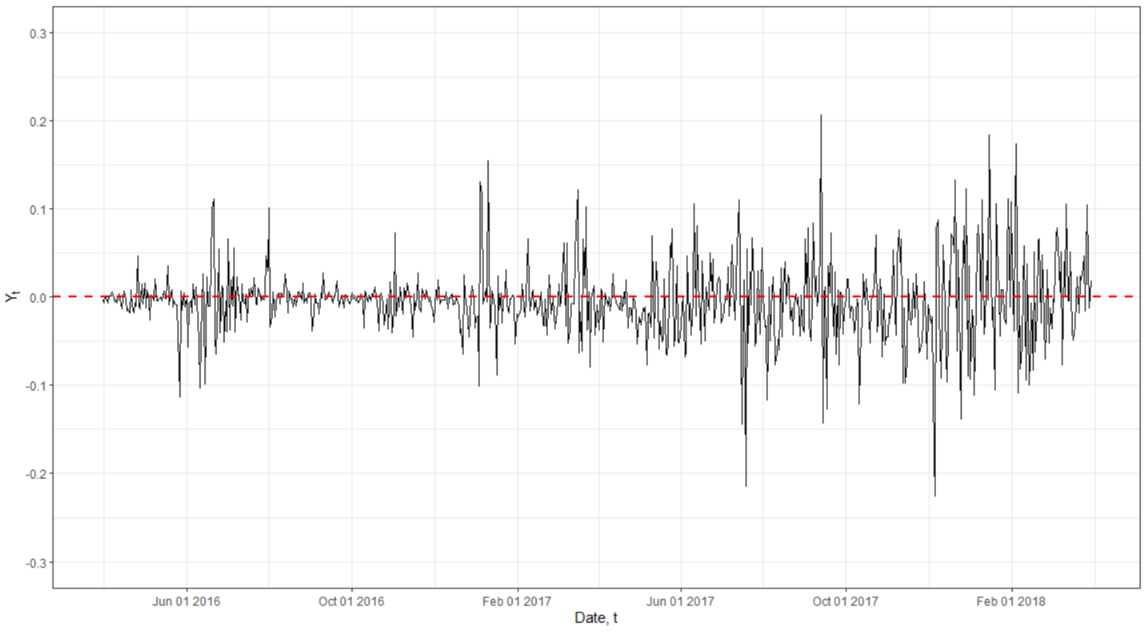


Figure 2: Time plot of USD/BTC daily log returns.

The primary aim of this mini-dissertation will entail deriving the properties of the hybrid ARMA-GARCH and estimating its parameters. The ARMA-GARCH model can essentially be described as an ARMA model with GARCH innovations. In Chapter 2 the necessary and preliminary results that will be used throughout the study will be given and we will present and give a detailed discussion on the ARMA-GARCH model. We will then apply the ARMA-GARCH modelling techniques discussed in Chapter 2 on a sample of Bitcoin returns with a focus on fitting the most appropriate model in order to predict future outcomes. We will then discuss linear regression models with either GARCH errors or ARMA-GARCH errors and derive the asymptotic theory required to estimate in particular, a linear regression with ARMA-GARCH errors. We will make use of preliminary test estimators and linear shrinkage test estimators to determine the best parameter estimates of the model derived in Chapter 4. These estimation procedures will then be discussed and derived in Chapter 5 and then applied to a numerical example. Lastly, in Chapter 6 the conclusion will be given and further comments will be made on any possible research extensions.

2 The ARMA-GARCH model

In this chapter, an introduction of the workings of some financial time series model will be provided. The preliminary results necessary will be defined first. Then in the sections that follow; the ARMA process, GARCH model and ARMA-GARCH model will be discussed and their properties defined.

2.1 Preliminary results

The following statistical results given will form the basis of our study, of the time series models discussed.

The primary goal of time series analysis is to bring about an understanding of the given data by building reliable models that explain the underlying stochastic process. With this in mind, the analysis of these models ultimately allows us to maximise our gains through the best possible predictions.

Throughout this study, we will consider the time series analysis of financial data. Let us define P_t, for $t = \ldots, -1, 0, 1, \ldots$ as the price of a financial asset (e.g. shares, stock indices, interest rates and other commodities etc.) at a point in time, t. As previously mentioned, returns series are preferred over their price series counterpart and therefore analysed due to their stable nature and estimable statistical characteristics. The log returns are also scale-free and thus allow us to compare different assets or commodities.

Definition 2.1.1. Let Y_t be a random variable defined as the log return on P_t given by

$$\begin{aligned} Y_t &= log P_t - log P_{t-1} \\ &= log\left(\frac{P_t}{P_{t-1}}\right) \\ &= log\left(1 + \frac{P_t - P_{t-1}}{P_{t-1}}\right). \end{aligned}$$

We previously defined the discrete random variable Y as the logarithmic changes of P (i.e. the price of an financial asset or commodity). Wold [61] theorised that the continuous state, discrete stochastic time series process $\{Y_t\}$ taken at any discrete time point, t is, in general, an amalgam of two elements; a deterministic and a nondeterministic distribution defined below.

Definition 2.1.2. The general decomposition of discrete random variable is Y_t is defined as follows

$$Y_t = m_t + \varepsilon_t. \tag{1}$$

$$\varepsilon_t = \sigma_t \eta_t. \tag{2}$$

where $\{m_t\}$ and $\{\varepsilon_t\}$ represent the deterministic and nondeterministic processes respectively, defined relative to a past information set Ω, inclusive of all relevant information up to time period t. The nondeterminitsic element ε_t, is driven by η_t (white noise) i.i.d. random variables such that $\eta_t \sim N(0,1)$. It then follows that the mean and variance of Y_t conditional on Ω_{t-1} respectively, are defined as

$$m_t = E\left[Y_t|\Omega_{t-1}\right]. \tag{3}$$

$$\begin{aligned}\sigma_t^2 &= Var\left[Y_t|\Omega_{t-1}\right] \\ &= E\left[\varepsilon_t^2|\Omega_{t-1}\right].\end{aligned} \tag{4}$$

where Ω_{t-1} denotes the information set as of time $t-1$ and $E\left[\varepsilon_t|\Omega_{t-1}\right] = 0$.

The following definitions given below are as given in Hamilton [26].

The basic foundation of all the time series models that will be described are built upon the white noise process defined below.

Definition 2.1.3. Let $\{\varepsilon_t\}_{t=-\infty}^{\infty}$ be a sequence of independent and identically distributed (i.i.d.) random variables such that

$$E(\varepsilon_t) = 0. \tag{5}$$

and

$$E\left(\varepsilon_t^2\right) = \sigma_\varepsilon^2. \tag{6}$$

This is called a white noise or innovation process denoted by $\varepsilon_t \sim i.i.d.\, N\left(0, \sigma_\varepsilon^2\right)$ where ε_t, $\forall\, t \in \mathbb{Z}$ are uncorrelated.

Definition 2.1.4. The pth order autoregressive (AR) model of the log return Y_t, denoted by $AR(p)$ is a linearly dependent polynomial of p of its previous terms (i.e. lags) and given as follows

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t, \qquad \forall\, t \in \mathbb{Z}. \tag{7}$$

or equivalently,

$$\left(1+\sum_{i=1}^{p}\phi_i L^i\right) Y_t \quad = \quad c+\varepsilon_t, \qquad \forall\, t \in \mathbb{Z}. \tag{8}$$

where c could be any constant, $\phi_1, \phi_2, \ldots, \phi_p$ are the regression coefficients and ε_t is the innovation process defined in Definition 2.1.3. and L is the lag or back shift operator such that $L^j y_t = y_{t-j}$.

Definition 2.1.5. The qth order moving average (MA) model of the log return Y_t, denoted by $MA\,(q)$ is a linear combination of q of the most recent innovations characterised by

$$Y_t \quad = \quad c+\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}+\ldots+\theta_q\varepsilon_{t-q}, \qquad \forall\, t \in \mathbb{Z}. \tag{9}$$

or equivalently,

$$Y_t \quad = \quad c+\left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t, \qquad \forall\, t \in \mathbb{Z}. \tag{10}$$

where c could be any constant, $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients of the corresponding innovations defined in Definition 2.1.2. and L is the lag or back shift operator such that $L^j y_t = y_{t-j}$.

We now go on to define the ARCH and GARCH models below for which the moments of the innovation process are conditional on past innovations. The ARCH model is based on a AR representation of the conditional variance and describes the process evolution of the conditional mean and variance.

Definition 2.1.6. A process $\varepsilon_t = \sigma_t \eta_t$ is said to be a $ARCH\,(p)$ process if it assumes the following conditions

(i)

$$E\left(\varepsilon_t \middle|\, \varepsilon_u,\, u<t\right) \quad = \quad 0, \qquad \forall\, t \in \mathbb{Z}. \tag{11}$$

(ii) Nonnegative constants exist such that $\alpha_0 > 0$ and $\alpha_i \geq 0,\ i = 1, 2, \ldots, p$ such that

$$\begin{aligned} Var\left(\varepsilon_t \middle|\, \varepsilon_u,\, u<t\right) &= \sigma_t^2 \\ &= \alpha_0+\sum_{i=1}^{p}\alpha_i\varepsilon_{t-i}^2, \qquad \forall\, t \in \mathbb{Z}. \end{aligned} \tag{12}$$

where η_t is a sequence of i.i.d. random variables such that $\eta_t \sim N\,(0,1)$.

2.2 ARMA model

The autoregressive moving average (ARMA) model is deemed the most widely used time series model because of its parsimony. It is an amalgam of the AR model and MA model. The ARMA model can better capture the behaviour of financial data compared its AR model or MA model counterparts. The AR model makes the effort to explain the momentum and mean reversion effects often observed in financial trading markets data whilst the MA model tries to capture the shock effects, such as unexpected events that affect the process being observed. ARMA models combine the features of both the AR and MA model. As great as this may sound, ARMA models do not have the ability to capture the volatility clustering of financial data which will be resolved by the GARCH process discussed in the next section.

Definition 2.2.1. The autoregressive moving average (ARMA) model of the log return Y_t , is a second-order stationary process with parameters p and q is a mixture of the $AR(p)$ and $MA(q)$ polynomials defined in (7) and (9) respectively. It is given as follows that

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}, \qquad \forall\, t \in \mathbb{Z}. \tag{13}$$

or equivalently,

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) Y_t = c + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t, \qquad \forall\, t \in \mathbb{Z}. \tag{14}$$

where c could be any constant, $\phi_1, \phi_2, \ldots, \phi_p$ are the regression coefficients, $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients of the corresponding innovations, $\{\varepsilon_t\}_{t=-\infty}^{\infty}$ is white noise as defined in Definition 2.1.3. and L is the referred to as the lag operator. The ARMA model is denoted $ARMA(p, q)$ where $p, q \in \mathbb{Z}$.
It should be noted that the roots of

$$\theta(z) \quad = \quad 1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q = 0. \tag{15}$$

lie outside the unit circle. Only the coefficients $\theta_1, \theta_2, \ldots, \theta_q$ (i.e. the parameters of the corresponding innovations) determine the invertibility of the process in (13) and (14) given that the assumption (15) holds.

It should also be noted that the roots of

$$\phi(z) \quad = \quad 1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p = 0. \tag{16}$$

also lie outside the unit circle. Given that the assumption in (16) holds, then the process in (13) and (14) is deemed covariance-stationary.

Definition 2.2.2. A process is said to be covariance-stationary if

$$E\left(Y_t\right) \quad = \quad \mu, \qquad \forall\, t,\, j. \tag{17}$$

the mean of the process is constant and thus independent of time and

$$E\left[\left(Y_t-\mu\right)\left(Y_{t-j}-\mu\right)\right] \quad = \quad \gamma_j, \qquad \forall\, t,\, j. \tag{18}$$

the autocovariances are independent of time.
This corresponds to the first and second order moments being independent of time.

Definition 2.2.3. The ARMA model defined in (14) can be re-expressed as the following

$$Y_t \quad = \quad \mu+\Psi\left(L\right)\varepsilon_t, \qquad \forall\, t \tag{19}$$

by dividing both sides of (14) by $\phi\left(L\right)=1-\phi_1 L-\phi_2 L^2-\ldots-\phi_p L^p$. We have that $\{\varepsilon_t\}_{t=-\infty}^{\infty}$ is white noise as defined in Definition 2.1.3,

$$\begin{aligned}
\Psi\left(L\right) &= \frac{1+\theta_1 L+\theta_2 L^2+\ldots+\theta_q L^q}{1-\phi_1 L-\phi_2 L^2-\ldots-\phi_p L^p} && (20)\\
&= \frac{\theta\left(L\right)}{\phi\left(L\right).}\\
\mu &= \frac{c}{1-\phi_1 L-\phi_2 L^2-\ldots-\phi_p L^p}. && (21)
\end{aligned}$$

and

$$\sum_{j=0}^{\infty}\left|\Psi_j\right| \quad < \quad \infty. \tag{22}$$

The process is described as ergodic for the mean provided that (22) holds. This simply means that the sample mean $\bar{y}$ converges in probability to the process mean given by (17) as the sample size $T\to\infty$.
The ARMA model is successful in capturing the movements of the conditional mean but assumes that the variance is constant and not conditioned on past information. This is a major drawback of the ARMA model as it fails to consider the possibility of the time-variant volatility which remains a main feature of financial time series. As can be seen in Figures 1 and 2 of the price and log returns of Bitcoin; there are several instances of extreme fluctuations and even calm periods which indicates that the data set has a non-constant variance also referred to as heteroscedasticity. It can also be seen in Figure 1 that the price of Bitcoin has an upward trend and then a downward trend which indicates non-stationarity.
The autoregressive integrated moving average (ARIMA) model defined below is a generalisation of the ARMA

model. The differencing step of one or more variables, corresponds to the "integrated" part of the model can be applied to a nonstationary time series; particularly those with a stochastic trend (i.e. it contains one or more unit roots). The idea of differencing is seen as a transformation that eliminates a process's non-stationarity (i.e. makes it stationary).

Definition 2.2.4. The log return series of Y_t that is integrated to the power d can be described by the autoregressive integrated moving average (ARIMA) model. The process is said to be an $ARIMA\,(p,d,q)$ and defined as follows

$$\left(1+\sum_{i=1}^{p}\phi_i L^i\right)\triangle^d Y_t = c+\left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t, \qquad \forall\, t\in\mathbb{Z}. \tag{23}$$

where c could be any constant, $\phi_1,\phi_2,\ldots,\phi_p$ are the regression coefficients, $\theta_1,\theta_2,\ldots,\theta_q$ are the coefficients of the corresponding innovations, $\{\varepsilon_t\}_{t=-\infty}^{\infty}$ is white noise as in Definition 2.1.3, L is the referred to as the lag operator and where $\triangle=(1-L)$ is the differencing step raised to a power d such that the parameters p, q and d are all nonnegative integers.

The metamorphosis of long-memory and fractional integration of ARIMA models were initially introduced through the works of Granger and Joyeux [25] in 1980 and later by Hosking [28] in 1981 respectively. Long memory is the idea that there is a slow decay in the dependence between observations a long span apart. In this case, the order of differencing, d of the ARIMA model may not be an integer but real (i.e. $|d|<1$). In particular, for values of $d>-\frac{1}{2}$ the process is invertible, when $d<\frac{1}{2}$ the process is stationary and when $d<1$ we have a mean-reverting while for $\frac{1}{2}<d<1$ the process is non-stationary. Fractional differencing allows the differencing operator of the ARIMA to be is raised to a fractional power with the use of an infinite binomial series expansion. When this occurs, the model is then referred to as the fractional differencing ARIMA model or formally known as the fractionally integrated autoregressive moving average (ARFIMA) model. AR models and non-stationary series have relatively long-memory and/or infinite memory meaning its current value has all its past innovations embedded in the series whereas any pure stationary ARIMA model as well as MA models are considered to have "short" or finite memory meaning that the innovations have a temporary effect.

2.3 GARCH model

Bollerslev [8] generalised the ARCH model (Definition 2.1.6.) introduced by Engle [17], referred to as the GARCH model. GARCH processes model financial time series while taking into account the important characteristics of a series. Their simplicity allows for an extended study of a model's statistical properties and probability. GARCH models express the conditional variance in the next time period as a linear combination of the past squared innovations which is the sum of the long-run average variance; the weighted average variance based on the current period; and lastly the weighted average of the new information set from the current period compared to the most recent squared residuals.

Definition 2.3.1. A process $\{Y_t\}$ is said to be a $GARCH\,(p,q)$ process if it satisfies the following conditions

(i)

$$Y_t \;=\; c+\varepsilon_t, \qquad \forall\, t\in\mathbb{Z}. \tag{24}$$

(ii)

$$\varepsilon_t \;=\; \sigma_t\eta_t. \tag{25}$$

(iii) There exists nonnegative constants, $\omega>0$; $\alpha_i\geq 0,\ i=1,2,\ldots,p$ and $\beta_j\geq 0,\ j=1,2,\ldots,q$ such that

$$\begin{aligned} Var\left(\varepsilon_t|\,\varepsilon_u,\,u<t\right) &= \sigma_t^2 \\ &= \omega+\sum_{i=1}^{p}\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^{q}\beta_j\sigma_{t-j}^2, \qquad \forall\, t\in\mathbb{Z}. \end{aligned} \tag{26}$$

where η_t is a sequence of i.i.d. random variables such that $\eta_t\sim N\left(0,1\right)$.

It should be noted that when $q=0$, the process defined above reduces to the $ARCH\,(p)$ process as defined in (11) and (12). The GARCH model has been well developed in the financial time series space because of its ability to capture some of skewness, excess kurtosis, volatility clustering and volatility mean reversion in which financial data exhibit, which are some of the properties inherited from the corresponding underlying process of Y_t.
We will consider the process Y_t with $GARCH\,(1,1)$ innovations because of its simplicity. The $GARCH\,(1,1)$ can easily be generalised to any $GARCH\,(p,q)$ with higher order of $p>1$ and/or $q>1$ which allow for more complex autocorrelation structures. However, the $GARCH\,(1,1)$ is typically used as it sufficiently explains financial time series well and higher order lags tend to be insignificant.

A simple description of the $GARCH\,(1,1)$ model as done by Reider [44] is given as follows.

Definition 2.3.2. A process $\{Y_t\}$ is said to be a univariate $GARCH\,(1,1)$ process is defined as follows

$$\begin{aligned} Y_t &= c+\varepsilon_t \\ \varepsilon_t &= \sigma_t\eta_t \\ \sigma_t^2 &= \omega+\alpha_1\varepsilon_{t-1}^2+\beta_1\sigma_{t-1}^2, \qquad \forall\, t\in\mathbb{Z}. \end{aligned} \tag{27}$$

where η_t is a sequence of i.i.d. random variables such that $\eta_t\sim N\,(0,1)$, the long-run average variance (i.e. unconditional variance of ε_t) of the $GARCH\,(1,1)$ process is given by $\frac{\omega}{1-\alpha_1-\beta_1}$ and the weights of the process are $(1-\alpha_1-\beta_1;\alpha_1;\beta_1)$ provided that $\alpha_1+\beta_1<1$ and $\omega>0$, $\alpha_1\geq 0$ and $\beta_1\geq 0$. We then rewrite the conditional variance in (27) using recursive substitution as follows

$$\begin{aligned}
\sigma_t^2 &= \omega+\alpha_1\varepsilon_{t-1}^2+\beta_1\sigma_{t-1}^2 \\
&= \omega+\alpha_1\varepsilon_{t-1}^2+\beta_1\left(\omega+\alpha_1\varepsilon_{t-2}^2+\beta_1\sigma_{t-2}^2\right) \\
&= \omega\left(1+\beta_1\right)+\alpha_1\varepsilon_{t-1}^2+\alpha_1\beta_1\varepsilon_{t-2}^2+\beta_1^2\sigma_{t-2}^2 \\
&= \omega\left(1+\beta_1\right)+\alpha_1\varepsilon_{t-1}^2+\alpha_1\beta_1\varepsilon_{t-2}^2+\beta_1^2\left(\omega+\alpha_1\varepsilon_{t-3}^2+\beta_1\sigma_{t-3}^2\right) \\
&\vdots \\
&= \omega\left(1+\beta_1+\beta_1^2+\ldots\right)+\alpha_1\varepsilon_{t-1}^2+\alpha_1\beta_1\varepsilon_{t-2}^2+\alpha_1\beta_1^2\varepsilon_{t-3}^2+\ldots \\
&= \omega\sum_{i=0}^{\infty}\beta_1^i+\alpha_1\sum_{j=1}^{\infty}\beta_1^{i-1}\varepsilon_{t-j}^2.
\end{aligned}$$

This reveals that the conditional variance, σ_t^2 has infinite long memory in terms of all the past squared residuals, ε_t^2, $\forall\, t$, which decay in coefficients. This property is the reason why GARCH models are considered better than their ARCH model counterparts.

As previously mentioned, there are complex alternatives to the standard GARCH model described above that would capture the different characteristics of financial time series. However, an in-depth analysis into the characteristics of the data provided has to be done, as well as into the different GARCH model types. For the sake of simplicity, it would be sufficient to consider the standard GARCH model with a different conditional distribution, η_t. In the application of GARCH modelling, the conditional distribution of the innovations is typically assumed to be normally distributed (i.e. $\eta_t \sim N\left(0,1\right)$) by default. Although in many cases of observed financial time series, Gaussian law is not exhibited. For instance, the observed returns series may, in fact, be skew and the kurtosis found is usually much greater than that implied by the normal GARCH model. Both measures are used in the empirical analysis to check possible deviations from normality such that, for any normally distributed variable, Y_t, with an unconditional mean $E\left[Y_t\right]=\mu_y$ and unconditional variance $Var\left[Y_t\right]=\sigma_y^2$, the skewness and kurtosis are defined as follows

$$\begin{aligned}
Skew\left[Y_t\right] &= E\left[\left(\frac{Y_t-\mu_y}{\sigma_y}\right)^3\right] \\
&= \frac{E\left(Y_t-\mu_y\right)^3}{\left(Var\left[Y_t\right]\right)^{\frac{3}{2}}} \\
&= \frac{E\left(Y_t-\mu_y\right)^3}{\sigma_y^3} \\
&= 0.
\end{aligned}$$

and

$$\begin{aligned}
Kurtosis\left[Y_t\right] &= E\left[\left(\frac{Y_t-\mu_y}{\sigma_y}\right)^4\right] \\
&= \frac{E\left(Y_t-\mu_y\right)^4}{\sigma_y^4} \\
&= 3.
\end{aligned}$$

We, therefore, extend upon the classical econometric approach by testing and using non-Gaussian distributions to try and better explain the time series. We will explore the different types of conditional distributions that the univariate standard GARCH can assume and will consider these in our analysis in the following chapter.

Definition 2.3.3. As stated in the *rugarch* [22] manual, the density function of a GARCH model

$$f\left(Y_t|\alpha_t\right) = f\left(Y_t|\omega,\, m_t,\, \sigma_t^2\right). \tag{28}$$

can be expressed in terms of its location and scale parameters which are normalised to give a mean of zero and a unit variance (i.e. $\eta_t \sim N\left(0,1\right)$) where the conditional mean and conditional variance are as defined in (3) and (4) respectively with the intercept ω, which denotes the remaining parameters of the distribution possibly including a shape and skewness parameter. The innovations of the process are scaled with the use of the conditional mean and variance such that the scaled version of the conditional distribution $\eta_t = \frac{\varepsilon_t}{\sigma_t}$, is given by

$$z_t = \frac{Y_t - m_t}{\sigma_t}. \tag{29}$$

which has a conditional density given by

$$g\left(z|\omega\right) = \frac{d}{dz}P\left(z_t < z|\omega\right). \tag{30}$$

It then follows that from (28) and (30) that

$$f\left(Y_t|\alpha_t\right) = \frac{1}{\sigma_t}g\left(z_t|\omega\right). \tag{31}$$

We will now discuss the different types of conditional density functions used to in GARCH modelling and give a comprehensive summary of how they are standardised in order to be used to better suit a given data set. It is vital that the conditional distribution be self-decomposable whilst still keep its linearity.

The following distribution defined below is as given by de Moivre [15]. and Gauss [20].

The Normal Distribution. A random variable Y that is normally distributed such that $N\left(\mu,\sigma^2\right)$ where the variance is constant and has the following density function

$$f\left(Y\right) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{Y-\mu}{\sigma}\right)^2}.$$

It follows that the residuals $\varepsilon = y - \mu$, can be standardised by the standard deviation of the process σ, which have a standard normal distribution with density function

$$\begin{aligned} f\left(\frac{Y-\mu}{\sigma}\right) &= \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}z^2} \\ &= \frac{1}{\sigma}f\left(z\right). \end{aligned} \tag{32}$$

This is the default conditional distribution of the GARCH innovations (i.e. $z_t = \eta_t \sim N(0,1)$).

The following distribution defined was developed by Gosset [24].

The Student-t Distribution. The Student-t distribution can be used as an alternative to the normal distribution as a conditional distribution of the GARCH innovations. This was first realised by Bollerslev [9] as it takes into account the underlying excess kurtosis in the data.
The random variable Y is Student-t distributed with μ the location parameter, β the scale parameter and $\nu > 2$ is the shape parameter. Then it follows that the density function of Y is given by

$$f(Y) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\beta\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{(y-\mu)^2}{\beta\nu}\right)^{-\left(\frac{\nu+1}{2}\right)}$$

where Γ is the gamma function. It should be noted that the location parameter μ is both the mean and the mode of the distribution.
The standardised Student-t density function is given by

$$\begin{aligned} f\left(\frac{Y-\mu}{\sigma}\right) &= \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sigma\sqrt{\pi(\nu-2)}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{z^2}{(\nu-2)}\right)^{-\left(\frac{\nu+1}{2}\right)} \\ &= \frac{1}{\sigma\sqrt{(\nu-2)}B\left(\frac{1}{2},\frac{\nu}{2}\right)}\left(1+\frac{z^2}{(\nu-2)}\right)^{-\left(\frac{\nu+1}{2}\right)} \\ &= \frac{1}{\sigma}f(z). \end{aligned} \tag{33}$$

where $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ represents the Beta function. The result above is obtained by substitution of the expression of the scale parameter β, in terms of the shape parameter ν. Since the variance of Y is standardised such that

$$Var(Y) = \frac{\beta\nu}{(\nu-2)} = 1.$$

$$\therefore \beta = \frac{\nu-2}{\nu}.$$

Furthermore, since the Student-t distribution is symmetric as with the normal distribution but allows for excess kurtosis (more specifically leptokurtic i.e. heavier tails), the skewness and kurtosis are given by

$$\begin{aligned} Skew(Y) &= 0. \\ Kurtosis(Y) &= \frac{6}{\nu-4}+3, \qquad \nu > 4. \end{aligned}$$

The Generalised Error Distribution. We will now consider another distribution belonging to the exponential family. The generalised error distribution (GED) often referred to as the exponential power distribution was first introduced by Subbotin [52] and then later suggested by Nelson [38] as a conditional distribution in terms of GARCH processes.

Let Y be a random variable being GED such that μ is the location parameter, β the scale parameter and $\nu > 0$ is the shape parameter, then the conditional density of Y is defined as follows

$$f(X) = \frac{\nu}{2^{1+\frac{1}{\nu}}\beta\Gamma\left(\frac{1}{\nu}\right)} e^{-\frac{1}{2}\left|\frac{Y-\mu}{\beta}\right|^{\nu}}.$$

This distribution shares the characteristic of symmetry like the normal and Student-t distributions. It is particularly special in that the mean, median and mode are all equal (i.e. μ). Varying the shape parameter ν reveals that when $\nu \to \infty$, the density tends to uniformity and we get a flatter distribution as $\nu \to 0$. It should also be noted that the Normal and Laplace distributions can be determined from the GED when $\nu = 2$ and $\nu = 1$, respectively.

All the odd order central moments are zero as with the skewness (i.e third central moment) and the even order central moments can be computed such that the second and fourth moments of the distribution are given by

$$\begin{aligned} Var(Y) &= \left(\beta 2^{\frac{1}{\nu}}\right)^2 \frac{\Gamma\left(\frac{3}{\nu}\right)}{\Gamma\left(\frac{1}{\nu}\right)}. \\ Kurtosis(Y) &= \frac{\Gamma\left(\frac{5}{\nu}\right)\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)\Gamma\left(\frac{3}{\nu}\right)}. \end{aligned}$$

When Y has a unit variance we can determine the scale parameter which is given by

$$\beta = \left(\frac{2^{-\left(\frac{2}{\nu}\right)}\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)}\right)^{\frac{1}{2}}.$$

Therefore the standardised scaled density function can be expressed as

$$f\left(\frac{Y-\mu}{\sigma}\right) = \frac{\nu}{\sigma 2^{1+\frac{1}{\nu}}\left(\frac{2^{-\left(\frac{2}{\nu}\right)}\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)}\right)^{\frac{1}{2}}\Gamma\left(\frac{1}{\nu}\right)} e^{-\frac{1}{2}\left|\left(\frac{2^{-\left(\frac{2}{\nu}\right)}\Gamma\left(\frac{1}{\nu}\right)}{\Gamma\left(\frac{3}{\nu}\right)}\right)^{\frac{1}{2}} z\right|^{\nu}}. \tag{34}$$

Skewed Distributions. Many of the conditional distributions described above are continuous unimodal and symmetric. Skewness can be introduced into the distribution by way of an inverse scaled factor on each side of the mode. This general approach was introduced by Fernandez and Steel [18] who stated that the density function of the random variable z, conditional on the skewness parameter ξ can be written as

$$f(z|\xi) = \frac{2}{\xi + \frac{1}{\xi}}\left[f(\xi z) H(-z) + f\left(\frac{z}{\xi}\right) H(z)\right]. \tag{35}$$

where $\xi \in (0, \infty)$ which is the shape parameter describing the degree of asymmetry such that in the instance that $\xi = 1$, the density function reduces to $f(z)$, the standardised function and $H(z) = \frac{(1+sign(z))}{2}$ is known as the Heaviside unit step function.

In order to obtain central order moments, the r-th absolute function, M_r defined below is used

$$\begin{aligned} M_r &= E\left(|X^r|\right) \\ &= 2\int_0^\infty x^r f\left(y\right) dy \end{aligned}$$

where $f\left(x\right)$ is on the positive real line. Then mean and variance are then given by

$$\begin{aligned} E\left(Y\right) &= \mu_\xi \\ &= M_1\left(\xi - \frac{1}{\xi}\right). \\ Var\left(Y\right) &= \sigma_\xi^2 \\ &= \left(M_2 - M_1^2\right)\left(\xi^2 - \frac{1}{\xi^2}\right) + 2M_1^2 - M_2. \end{aligned}$$

The scaled or standard version of the skewed distributions form is a simply a re-parametisation of (35) so that the mean is zero and we have unit variance. The standardised skewed distribution probability function is obtained using (35) , the moment conditions given above and the transformation of the variable $X \to z^* = \frac{x-\mu}{\sigma}$ which gives

$$f\left(z^*|\xi\theta\right) = \frac{2\sigma}{\xi + \frac{1}{\xi}} f^*\left(z_{new}|\theta\right).$$

where $f^*\left(z|\theta\right)$ represents any of the standardised continuous symmetric unimodal distribution functions described in (32), (33), (34) and (35). We also have that $z_{new} = \xi^{sign(\sigma_\xi z + \mu_\xi)}\left(\sigma_\xi z + \mu_\xi\right)$, $\xi \in (0, \infty)$ and that θ represents the optional set of shape parameters particularly for the the GED and Student-t distribution in the case of higher order moments.

The Generalised Hyperbolic Distribution family. The family of generalised hyperbolic (GHYP) distributions was first introduced by Barndorff-Nielsen [5] in 1977. The GHYP distribution is a normal variance-mean mixture which means it has semi-heavy tails. There have been various extensions of the distribution family over the years with a particular focus on financial applications. It was Barndorff-Nielsen and Blaesild [6] in 1981 who discussed the first four moments, skewness and kurtosis of the distribution. It is rather complicated to define an explicit form of the distribution's moments as they are functions of all the five parameters. Although a recursive method was used by Scott et al. [50] to obtain the moments of any order in a simplified manner.
The random variable X is said to have a generalised hyperbolic distribution with a probability function defined as

$$\begin{aligned} f\left(Y|\lambda,\, \alpha,\, \beta,\, \delta,\, \mu\right) &= \frac{\left(\alpha^2 - \beta^2\right)^{\frac{\lambda}{2}}}{\sqrt{2\pi}\alpha^{\lambda-\frac{1}{2}} K_\lambda\left(\delta\sqrt{\alpha^2 - \beta^2}\right)} \left[\delta^2 + (x-\mu)^2\right]^{\frac{\left(\lambda - \frac{1}{2}\right)}{2}} K_{\lambda - \frac{1}{2}}\left(\alpha\sqrt{\delta^2 + (y-\mu)^2}\right) e^{[\beta(y-\mu)]} \\ &= a\left(\lambda,\, \alpha,\, \beta,\, \delta,\, \mu\right)\left[\delta^2 + (y-\mu)^2\right]^{\frac{\left(\lambda - \frac{1}{2}\right)}{2}} K_{\lambda - \frac{1}{2}}\left(\alpha\sqrt{\delta^2 + (y-\mu)^2}\right) e^{[\beta(y-\mu)]}. \qquad (36) \end{aligned}$$

where K_λ is the modified Bessel function of the the third kind of order λ. The location and scale are determine by μ and δ whilst the shape of the density is determined by α and β. We have that $\alpha,\, \mu \in \mathbb{R}$ and the remaining parameters are given as

$$\delta,\, |\beta| = \begin{cases} \delta \geq 0,\ |\beta| < \alpha & if\ \lambda > 0 \\ \delta > 0,\ |\beta| < \alpha & if\ \lambda = 0 \\ \delta > 0,\ |\beta| \leq \alpha & if\ \lambda < 0 \end{cases}$$

then we say that $Y \sim GHYP\,(\lambda,\, \alpha,\, \beta,\, \delta,\, \mu)$ (first parameterisation (α, β)).

The distribution has three alternative scale-invariant and location-invariant parameterisation forms that were discussed in a thesis presented by Prause [41]. We will focus on the second parameterisation (ζ, ρ) as this used in the *rugarch* [22] package in R [14] which then transforms those parameters into the first parameterisation. The second parameterisation parameters are defined in the following way

$$\begin{aligned} \zeta &= \delta\sqrt{\alpha^2 - \beta^2}. \\ \rho &= \frac{\beta}{\alpha}. \end{aligned}$$

Suppose that $Y \sim GHYP\,(\lambda,\, \alpha,\, \beta,\, \delta,\, \mu)$ then the r-th central moment of the GHYP distribution can be proved to be

$$\begin{aligned} M_r &= E\,(Y - \mu)^r \\ &= \sum_{s=\left[\frac{(r+1)}{2}\right]}^{r} \frac{r!}{(r-s)!\,(2s-r)!2^{r-s}} \beta^{2s-r} E\,(W^s) \\ &= \sum_{s=\left[\frac{(r+1)}{2}\right]}^{r} a_{r,s} \beta^{2s-r} E\,(W^s) \end{aligned}$$

using the the moments results of the standard normal distribution

$$E\left(Z^i\right) = \begin{cases} 0 & when\ i\ odd. \\ \frac{i!}{2^{\frac{i}{2}}\left(\frac{i}{2}\right)!} & when\ i\ even. \end{cases}$$

and where

$$\begin{aligned} E\,(W^s) &= \left(\frac{\delta^2}{\delta\sqrt{\alpha^2 - \beta^2}}\right)^{\lambda+s} \frac{K_{\lambda+s}\left(\delta\sqrt{\alpha^2-\beta^2}\right)}{K_\lambda\left(\delta\sqrt{\alpha^2-\beta^2}\right)} \\ &= \left(\frac{\delta^2}{\zeta}\right)^{\lambda+s} \frac{K_{\lambda+s}\,(\zeta)}{K_\lambda\,(\zeta)}. \end{aligned}$$

and

$$a_{r,s} = \begin{cases} 0 & s < \left[\frac{(r+1)}{2}\right]. \\ \frac{r!}{(r-s)!(2s-r)!2^{r-s}} & \left[\frac{(r+1)}{2}\right] < s < r. \end{cases} \quad where\, s = 1, 2, \ldots k\; and\, r = 1, 2, \ldots$$

Then it follows that the mean and variance of the GHYP distribution is given by

$$\begin{aligned} E(Y) &= \mu + \left(\frac{\delta^2}{\zeta}\right)\beta\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}. \\ Var(Y) &= \left(\frac{\delta^2}{\zeta}\right)\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)} + \left(\frac{\delta^2}{\zeta}\right)^2\beta^2\left[\frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)} + \left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^2\right]. \end{aligned}$$

and then the skewness and kurtosis are expressed as follows

$$\begin{aligned} Skew(Y) &= [Var(Y)]^{-\frac{3}{2}}\left[\left(\frac{\delta^2\beta}{\zeta}\right)^3\left(\frac{K_{\lambda+3}(\zeta)}{K_\lambda(\zeta)} - \frac{3K_{\lambda+2}(\zeta)K_{\lambda+1}(\zeta)}{(K_\lambda(\zeta))^2} + 2\left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^3\right)\right] \\ &\quad + [Var(Y)]^{-\frac{3}{2}}\left[3\left(\frac{\delta^2}{\zeta}\right)^2\beta\left(\frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)} - \left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^2\right)\right]. \\ Kurtosis(Y) &= -3 + [Var(Y)]^{-2}\left[\left(\frac{\delta^2\beta}{\zeta}\right)^4\left(\frac{K_{\lambda+4}(\zeta)}{K_\lambda(\zeta)} - \frac{4K_{\lambda+3}(\zeta)K_{\lambda+1}(\zeta)}{(K_\lambda(\zeta))^2} + \frac{6K_{\lambda+2}(\zeta)(K_{\lambda+1}(\zeta))^2}{(K_\lambda(\zeta))^3} + 3\left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^4\right)\right] \\ &\quad + [Var(Y)]^{-2}\left[\left(\frac{\delta^2}{\zeta}\right)^3\beta^2\left(\frac{6K_{\lambda+3}(\zeta)}{K_\lambda(\zeta)} - \frac{12K_{\lambda+2}(\zeta)K_{\lambda+1}(\zeta)}{(K_\lambda(\zeta))^2} + 6\left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^3 + 3\left(\frac{K_{\lambda+1}(\zeta)}{K_\lambda(\zeta)}\right)^4\right)\right] \\ &\quad + [Var(Y)]^{-2}\left[3\left(\frac{\delta^2}{\zeta}\right)^2\frac{K_{\lambda+2}(\zeta)}{K_\lambda(\zeta)}\right]. \end{aligned}$$

It then follows that the standardised GHYP distribution density can be expressed as

$$\begin{aligned} f\left(\frac{Y-\mu}{\sigma}|\zeta,\,\rho\right) &= \frac{1}{\sigma}f(z|\zeta,\,\rho) \\ &= \frac{1}{\sigma}f\left(z|\tilde{\lambda},\,\tilde{\alpha},\,\tilde{\beta},\,\tilde{\delta},\,\tilde{\mu}\right). \end{aligned}$$

where z is scaled version of the random variable $Y \sim GHYP(\zeta, \rho)$ to have zero mean and unit variance by way of transforming it from the second parameterization to the first.

It should be noted that for symmetric distributions $\beta = \tilde{\beta} = \rho = 0$ and other special distributions can be obtained from the GHYP such the Normal and Normal Inverse Gaussian (NIG) distribution in the case where the shape parameter $\lambda = 1$ or $\lambda = -0.5$ respectively.

The Generalised Hyperbolic Skew Student-*t* Distribution. In the previous section we discussed the GHYP distribution and its properties. In particular we looked at symmetric GHYP distributions. There has been further research into non-symmetric of the skewed generalised hyperbolic distributions. It was previously noted that GHYP distributions are symmetric when $\beta = \tilde{\beta} = \rho = 0$ and conversely will be skewed when $\beta \neq 0$. We focus on the generalised hyperbolic skew Student's t-distribution which was not well-known until Aas and Haff [1] popularised it. It is a heavier tailed distribution than that of the normal distribution and it also has quite a special property in how it behaves; it is the only one of its kind in the subclass of GHYP family in that one tail bears exponential traits while the other tail exhibits polynomial behaviour compared to the symmetric Student's t-distribution that has polynomial decaying tails. As previously defined in (36), the density of the GHYP is given by

$$f\left(Y|\lambda,\, \alpha,\, \beta,\, \delta,\, \mu\right) = \frac{\left(\alpha^2-\beta^2\right)^{\frac{\lambda}{2}}}{\sqrt{2\pi}\alpha^{\lambda-\frac{1}{2}}K_{\lambda}\left(\delta\sqrt{\alpha^2-\beta^2}\right)}\left[\delta^2+\left(y-\mu\right)^2\right]^{\frac{\left(\lambda-\frac{1}{2}\right)}{2}}K_{\lambda-\frac{1}{2}}\left(\alpha\sqrt{\delta^2+\left(y-\mu\right)^2}\right)e^{\left[\beta\left(y-\mu\right)\right]}.$$

The GHYP skew Student's t-distribution is defined such that $\lambda=-\frac{\nu}{2}$ where $\nu>0$ determines the shape of the distribution and the parameter $\alpha\to|\beta|$ and $\beta\in\mathbb{R}$. It should be noted that since ν is the distribution's shape parameter, for the variance to be finite and the skewness and kurtosis to exist $\nu>4$, $\nu>6$ and $\nu>8$ respectively opposed to it symmetric counterpart. This is a limiting case of the GHYP distribution
Let Y be a random variable that has a generalised hyperbolic skew Student's t-distribution then the probability density function is given by

$$f\left(Y|\nu,\, \beta,\, \delta,\, \mu\right) = \frac{2^{\frac{1-\nu}{2}}\left|\beta\right|^{\frac{\nu+1}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\delta\sqrt{\alpha^2-\beta^2}\right)^{\frac{\nu+1}{2}}}K_{\frac{\nu+1}{2}}\left(\sqrt{\beta^2\left(\delta^2+\left(x-\mu\right)^2\right)}\right)e^{\left[\beta\left(y-\mu\right)\right]}. \quad (37)$$

The mean and variance of the distribution can then be standardized to give a $Z\sim N\left(0,1\right)$ such that

$$\begin{aligned} E\left(Y\right) &= \mu+\frac{\beta\delta^2}{\nu-2}=0. \\ Var\left(Y\right) &= \frac{2\beta^2\delta^4}{\left(\nu-2\right)^2\left(\nu-4\right)}+\frac{\delta^2}{\nu-2}=1. \end{aligned}$$

The normal mixture of the distribution makes the derivation of the skewness and kurtosis explicit forms complex to obtain as they are also limited to exist when $\nu>6$ and $\nu>8$ respectively. The expressions can be seen below.

$$\begin{aligned} Skew\left(Y\right) &= \frac{2\left(\nu-4\right)^{\frac{1}{2}}\beta\delta}{\left[2\beta^2\delta^2+\left(\nu-2\right)\left(\nu-4\right)\right]^{\frac{3}{2}}}\left[3\left(\nu-2\right)+\frac{8\beta^2\delta^2}{\left(\nu-6\right)}\right]. \\ Kurtosis\left(Y\right) &= \frac{6}{\left[2\beta^2\delta^2+\left(\nu-2\right)\left(\nu-4\right)\right]^2}\left[\left(\nu-2\right)^2\left(\nu-4\right)+\frac{16\beta^4\delta^2\left(\nu-2\right)\left(\nu-4\right)}{\left(\nu-6\right)}+\frac{8\beta^4\delta^4\left(5\nu-22\right)}{\left(\nu-6\right)\left(\nu-8\right)}\right]. \end{aligned}$$

As we recall there are four scale-invariant and location-invariant parameterisation forms presented by Prause [41]. In our analysis the fourth parameterisation $\left(\bar{\beta}=\beta\delta,\, \bar{\alpha}=\alpha\delta\right)$ is used and reparameterised to $\left(\bar{\beta},\nu\right)$ as used in the *rugarch* [22] package in R [14] which then transforms those parameters into the first parameterisation $\left(\alpha,\beta\right)$.
The standardised density of the GHYP skew Student's t-distribution as stated by Barndorff-Nielsen and Blaesild [6] is a normal variance mixture with the generalised inverse Gaussian (GIG) distribution as the mixing component is given by

$$\begin{aligned} f\left(\frac{Y-\mu}{\sigma}\right) &= \frac{1}{\sigma}f\left(z|\lambda,\, \alpha,\, \beta,\, \delta\right) \\ &= \left(\frac{\sqrt{\alpha^2-\beta^2}}{\delta}\right)^{\lambda}\frac{z^{\lambda-1}}{2\sigma K_{\lambda}\left(\delta\sqrt{\alpha^2-\beta^2}\right)}e^{\left\{-\frac{1}{2}\left[\delta^2z^{-1}+\left(\alpha^2-\beta^2\right)z\right]\right\}} \end{aligned}$$

where $\lambda = -\frac{\nu}{2}$ and $\alpha \rightarrow |\beta|$.

Johnson's SU distribution. The last conditional distribution we will consider that the error distribution may assume is Johnson's S_u (JSU) distribution. It was first introduced by Johnson [30] in 1949. This is a four-parameter distribution defined such that ξ and $\lambda > 0$ are the location and scale parameters, and it consists of two shape parameters γ and $\delta > 0$. The probability density function is then given by

$$f\left(Y|\xi,\, \lambda,\, \gamma,\, \delta\right) \quad = \quad \frac{\delta}{\lambda\sqrt{2\pi}\sqrt{\left(\frac{y-\xi}{\lambda}\right)^2+1}} e^{\left\{-\frac{1}{2}\left[\gamma+\delta ln\left(\left(\frac{y-\xi}{\lambda}\right)+\sqrt{\left(\frac{y-\xi}{\lambda}\right)^2+1}\right)\right]^2\right\}}.$$

for $-\infty < y < +\infty$ where $z = \left(\frac{y-\xi}{\lambda}\right)$.
The JSU was then reparameterised by Rigby and Stasinopoulos [45] denoted by $JSU\left(\mu, \sigma, \nu, \tau\right)$ where μ is the location shift parameter, σ is the scale parameter, ν and τ determine the skewness and kurtosis respectively, and $z = \left(\frac{y-\xi}{\lambda}\right) = \nu + \tau \sinh^{-1}\left(\frac{y-\mu}{\sigma}\right) \sim N\left(0,1\right)$. The reparameterised probability density function is then given by

$$f\left(Y|\mu,\, \sigma,\, \nu,\, \tau\right) \quad = \quad \frac{\tau}{\sigma\sqrt{2\pi}\sqrt{\left(\frac{y-\mu}{\sigma}\right)^2+1}} e^{\left\{-\frac{1}{2}z^2\right\}}.$$

for $y \in (-\infty, +\infty)$ where $\mu,\, \nu \in (-\infty, +\infty)$ and $\sigma,\, \tau > 0$.
It should also be noted that this is a leptokurtic distribution and in a special case where $\tau \rightarrow \infty$, the normal distribution can be obtained.

2.4 ARMA-GARCH models

In this section, we will define and discuss the ARMA-GARCH model.

The standard GARCH process may indeed model financial time series well by capturing some of the excess skewness and kurtosis but it does have its downside. Since the introduction of this model, in the last three decades, extensions of this model have been developed to get better descriptions of financial data.

The ARMA-GARCH model is one of many extensions of the GARCH process with respect to the conditional mean. This dynamic model allows for both the conditional mean and conditional variance to be modelled by the ARMA process and GARCH process respectively.

Definition 2.4.1. Suppose that a set of observations $Y_1, Y_2, \ldots, Y_T$ of a log returns times series are generated by the ARMA model with its given errors generated by the GARCH process. It then follows that the $ARMA\left(p,q\right) - GARCH\left(m,n\right)$ model is defined as

(i)

$$\begin{aligned} Y_t &= m_t + \varepsilon_t \\ &= c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t, \qquad \forall t \in \mathbb{Z}. \end{aligned} \tag{38}$$

where $m_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$ is the conditional mean of the $ARMA\,(p,q) - GARCH\,(m,n)$ and ε_t satisfies the following

(ii)

$$\varepsilon_t = \sigma_t \eta_t. \tag{39}$$

(iii) There exists nonnegative constants ω, α_i, $i = 1, 2, \ldots, n$ and β_j, $j = 1, 2, \ldots, m$ such that the conditional volatility

$$\begin{aligned} Var\,(\varepsilon_t |\, \varepsilon_u,\, u < t) &= \sigma_t^2 \\ &= \omega + \sum_{i=1}^{n} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{m} \beta_j \sigma_{t-j}^2, \qquad \forall t \in \mathbb{Z}. \end{aligned} \tag{40}$$

where η_t is a sequence of i.i.d. random variables such that $\eta_t \sim N\,(0,1)$ and have a common density f.

The traditional ARMA model defined in (19) and (20) is usually preferred over the MA or AR models due to its parsimony. Although it is also assumed that noise process, ε_t, is assumed to be i.i.d. $N\left(0, \sigma_\varepsilon^2\right)$ with a zero mean and constant variance. This assumption has proven to be inadequate especially in the case of financial time series because we know that the squared price returns are autocorrelated; future variation is conditional on past variation; volatility clustering exists and financial data tends to be leptokurtic. Therefore by combining the GARCH process to model the innovations of an ARMA model, a better model can be obtained.
As previously discussed, the conditional distribution of the noise process may not necessarily be normal and therefore depending on the nature of the data, we may assume any one of the distributions mentioned in the previous section (i.e. GED, GHYP etc.)

The estimation of the ARCH-class family models have primarily been carried out using quasi-maximum likelihood estimators (QMLE). The asymptotic theory of QMLE of ARCH models was first studied and demonstrated by Weiss [59] in 1986. He proved that the MLE of ARCH models were both asymptotically normal and consistent and that non-normal data had finite fourth moments. However, his methods were not applicable to GARCH processes as certain conditions would be violated. It was only until the early 1990s that Lumsdaine [35] and later, Lee and Hansen [33] who made significant further developments in the asymptotic theory of QMLE for GARCH models. This was performed on mainly the $GARCH\,(1,1)$ and $IGARCH\,(1,1)$ processes. Since then, more literature has circulated in and around other GARCH model types. For instance, Francq and Zakoian [19] prove the consistency and asymptotic normality of pure GARCH and pure ARMA-GARCH processes using QMLE in the estimation of the parameters whereas Aknouche and Bibi [2] do the same for the periodic GARCH and periodic ARMA-GARCH and many others.

3 ARMA-GARCH Application of Bitcoin Data

In this chapter, a brief introduction to cryptocurrencies will be given. We will then apply our statistical and financial time series analysis techniques discussed in the previous chapter on a sample of Bitcoin data. We will document our results and verify that our predicted model is correct.

3.1 Cryptocurrencies

The concept of digitised assets and innovation of financial instruments has changed the way in which we view money and finance. In the last decade, the world has adopted a new financial system paradigm; the cryptocurrency. A cryptocurrency is defined as a decentralised digital currency, absent of government control and interference and has no fixed or physical underlying commodity to back it up; similar to that of a fiat currency, although not declared by legal tender by the government. Instead, cryptocurrencies are designed to be an exchange medium on a public network which is built, stored and secured by the generation of blockchains that are encrypted and controlled by the use of cryptography. A cryptographic platform or blockchains' use of encryption techniques, controls the supply of additional monetary units (e.g. Bitcoin) whilst these transactions are verified and stored electronically. Essentially, cryptocurrencies are based on the solution of cryptographic puzzles using blockchain technology to store and validate transactions on a public ledger. Cryptocurrencies are now not only a globalised but localised medium of exchange.

3.2 Bitcoin

Bitcoin was the first existing cryptocurrency introduced and operational in 2009 as a peer-to-peer electronic cash system which can be freely traded. A paper released in late 2008 authored by a person or group under the pseudonym Satoshi Nakamoto titled “Bitcoin: A peer-to-peer electronic cash system” was the initial introduction of this cryptocurrency technology. In recent years, Bitcoin has grown quite rapidly reaching a peak of nearly

US$18000 per Bitcoin unit in December 2017 but has since lost significant value. With over a thousand tradable cryptocurrencies to date, having a market value of about $260 billion, Bitcoin being the first cryptocurrency which also holds the majority of the cryptocurrency market capitalization; 109 billion USD as of July 2018. It has now been accepted by various businesses as a form of exchange and in some countries also a legalised form of payment (i.e. fiat currency).

Cryptocurrencies with Bitcoin, in particular, have sparked a lot of interest and attention due to their highly volatile nature compared to more traditional mediums of exchange caused by their susceptibility to speculation as they are not backed by an underlying asset. The price of such cryptocurrencies is partially due to the interest they have attracted. However, the sudden negative changes in the price of Bitcoin have been difficult to explain. This new craze has resulted in people investing their money blindly which hikes up the price. Because of its volatile nature, which has brought about a lot of uncertainty and in turn, its current and future standings could possibly revolutionise the economic and financial sectors forever or become obsolete. People's understanding is still limited around this technology with many people speculating that this is just another bubble with a soon approaching expiration date

We intend to fit an appropriate ARMA-GARCH model to the sample data displayed in Figures 1 and 2 and verify the estimated parameters of the model that can be used in forecasting.

3.3 Literature review

The birth of the cryptocurrency and its relative unpredictability has boosted further interest in research areas within the financial and economic spheres. Although due to its recent introduction, literature in this regard is still minimal, it is definitely a growing field of interest. It was only up until 2013 that interest started to emerge as more people started to grasp the concept of this new technological advancement. In the last two years, several articles have been written that examine Bitcoin's volatility and future outcomes. Some of the authors would include Polasik et al. [40]; Katsiampa [31]; and Chu et al. [12]. Katsiampa fitted and compared different GARCH models to Bitcoin data and found that the AR-CGARCH gives the optimal fit. He also gave mention of the highly speculative nature of the cryptocurrency which contributes to its extremely volatile nature. Vo and Xu [57] analysed the volatility of hourly Bitcoin log returns. They found that the best fitted model to the data was an $ARMA(1,2) - GARCH(2,2)$ with a TGARCH sub-model and a GHYP distribution as the underlying distribution. They also ran a credibility check of their estimated ARMA-GARCH model, comparing it against support vector machines (SVM) and neural networks (NN) and still concluded that the ARMA-GARCH specification better captured the movements of the Bitcoin data series. Their correlation analysis proved that Bitcoin is not affected by traditional financial markets under similar constraints.

There has been plenty of debate in regards to Bitcoin's currency characterisation. It cannot be characterised as a fiat currency as it is not backed by any governmental institution but rather controlled by it users; a peer-to-peer network. All the users are involved in the creation, trading and validation (i.e. process and check) of Bitcoin transactions.

Bitcoin's price is highly volatile and fluctuates based on investors willingness to pay for a Bitcoin at any point in time. This has raised some fear amongst notable financial figures. This decentralised system allows people to trade freely and globally in order to build a multi-billion dollar financial system. It was previously mentioned that Vo and Xu [57] proved that Bitcoin has no effects on the current, traditional financial systems. Authors such as Bouoiyour et al. [10] are in favour of Bitcoin and others such as Yermack [63] who are still speculative of this new concept and fear that this is just a bubble but it is inevitable that it has and will continue to change our trading practices.

A paper was released by Sovbetov [51] recently looks at both the short-run and long-run dynamic factors that contribute to the price of a cryptocurrency. These include internal factors such as supply and demand and its external factors being the overall cryptocurrency market, macro-finance and politics. The author analyses the top 5 cryptocurrencies in the market and indicates that the volatility of the cryptocurrency market has a significantly negative effect. It was also determined that attractiveness was a factor of significance while the control variable (i.e. S&P 500 stock index) being part of the traditional financial system had little to no effect on the price.

3.4 Data

Given this internet age, access to past and current Bitcoin (BTC) price and return data are plenty and can be found with ease online. However, most of these data sets only date back as far as 2014. These databases are considerably short term and it may be quite difficult to obtain meaningful results even with a thorough analysis. Although in our analysis will be performed on the most recent Bitcoin exchange markets data from the *crypto* [56] package in R [14] sourced from CoinMarketCap. We chose USD/BTC daily rates between the period of March, 31st, 2016 to March, 31st, 2018 which depicts the most volatility and highest price peak as Bitcoin gained increasing interest. There is a total of 731 observations of daily price trades. These are also displayed in Figure 1 and 2 in Chapter 1.

3.5 Empirical Analysis

We conducted an empirical analysis on the Bitcoin data described in the preceding section. We first took a look at the basic statistics of the daily prices data of Bitcoin. This revealed that the time series has a distribution that is skewed to the right and leptokurtic (i.e. evidence of a fat tail or heavy-tailed distribution). We then tested for stationarity using the Augmented Dickey-Fuller (ADF) test on the USD/BTC daily prices and it was found that at 5% significance level, we cannot reject the null hypothesis of a unit-root, therefore we conclude that the data is non-stationary. This is confirmed by the time plot displayed in Figure 1 which depicts an upward and downward trend of the time series. It is, therefore, necessary for us to difference the time series to obtain daily log returns as displayed in figure 2. The basic statistics reveal that the log returns time series is slightly skewed to the right and extremely leptokurtic, having a very high excess kurtosis. The ADF test for the log returns time series indicates stationarity since we reject the null hypothesis of a unit root with 95% confidence. The basic statistics and ADF test results for both the Bitcoin time series are reported in the following table.

Table 1: Basic statistics and ADF test results of Bitcoin price and log returns respectively.

Basic Statistics	Price (P_t)	Log returns (Y_t)
Minimum	416.730000	−0.225119
Maximum	19497.400000	0.207530
1st Quartile (Q1)	662.765000	−0.021965
Median (Q2)	1187.470000	−0.003625
3rd Quartile (Q3)	4568.667500	0.009133
Mean	3540.000874	−0.003829
Standard Deviation	4281.490410	0.043638
Skewness	1.638178	0.136596
(Excess) Kurtosis	1.850749	4.093399

ADF results	Price (P_t)	Log returns (Y_t)
Test statistic	−2.1649	−7.8737
p-value	0.5085	0.01

Henceforth, our analysis will be performed on the log returns series, Y_t. We run further tests to determine what the underlying distribution of the log returns is. These include the basic 2-sided t-test as well as tests under the assumption of normality such as the test for skewness, kurtosis and Jarque-Bera (JB) test. A Q-Q (quantile-quantile) plot is also drawn. If it is found that the hypothesis of normality is false then we will deduce in which family class it lies. The results of the tests mentioned were obtained with the use of the *normtest* [21] package in R [14] and will be given in the Table 2 below.

Table 2: Underlying distribution normality tests.

	Two Sided t-test	Skewness Test	Kurtosis Test	Jarque-Bera Normality Test
Test statistic	−2.3721	0.13688	7.1128	517.5
p-value	0.01794	0.1215	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$

From Table 2, we reject the null hypothesis that the mean of the log returns is zero and yet the mean ($\bar{x} = -0.003828$) is not significantly different from zero given the results of the t-test. The skewness normality test also fails to reject the null hypothesis and as a result suggests that the log-returns series is symmetric. The kurtosis test was then also performed; this revealed that our log returns series is indeed leptokurtic as we reject the null hypothesis of normality with 95% confidence. The notion that the log returns series is non-normal is reaffirmed by the JB-test the null hypothesis of normality was rejected even at a 1% level of significance. The Q-Q plot given in Figure 3 also validates our conclusion that the log returns are not normally distributed and have heavy tails. The skewness and kurtosis tests are extremely sensitive and the unusual results from the skewness test may be due to possible outliers and considering we are working with a small concentrated data set. This affects the symmetry of the distribution making it appear skew when it truly isn't. From the basic statistics in Table 1 also suggested a slight skewness to the left. From these results, we can conclude that the underlying or conditional distribution is not from the Gaussian family and therefore other family classes must be considered.

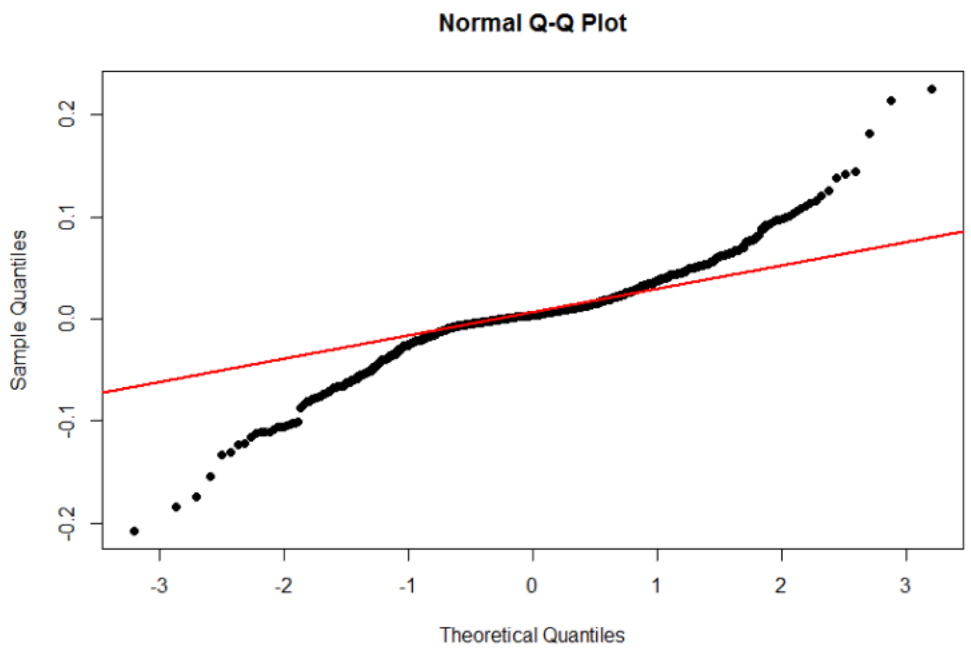


Figure 3: Q-Q plot of log returns distribution.

As has been shown, the sample of the log returns Bitcoin data exhibit attributes that lead us to conclude that it can be modelled by a non-Gaussian distribution. Chu et al. [13] in 2015 performed a thorough statistical analysis on the Bitcoin exchange rate by fitting fifteen different parametric distributions to the data and found that the GHYP distribution fits best. In our analysis, we fitted several symmetric and non-symmetric univariate distributions to log returns of the sample data. The *QRM* [39] package in R [14] was used to fit the Normal, Student-t, hyperbolic and NIG distributions. The GED and GHYP distributions were fitted to the log returns using the R [14] packages *fGarch* [62] and *ghyp* [34] respectively. In terms of estimation of the parameters of the models fitted to the data, the Maximum Likelihood Estimation (MLE) was the method of choice for its simplicity, reliability of results and expeditious convergency. We compared each of the models using the Akaike Information Criterion (AIC) defined as

$$AIC \quad = \quad -2\ln L(\Theta) + 2K.$$

where $\ln L(\Theta)$ represents the maximum log-likelihood, K being the number of parameters estimated. Information criterion like the AIC, measure the loss of information and are used to determine a best fitted model when their information criterion values are compared. Therefore, a model with smaller information criterion is preferred as it indicates a lower degree of information lost in comparison to other models and hence, the better fit.

The model with the smallest AIC was then selected as the best fitted distribution model to the log returns of our data. Given in the following table are the model fitting results as follows

Table 3: Model fitting results (AIC) of different univariate parametric distributions fitted to the log returns of BTC/USD exchange.

Distributions	$\ln L(\Theta)$	AIC
Normal distribution	1250.545	−2497.09
Student-t distribution	1350.865	−2695.73
GED	1377.069	−2748.139
(symmetric) Hyperbolic distribution	1273.354	−2540.709
(skew) Hyperbolic distribution	1272.241	−2538.481
(symmetric) NIG distribution	1363.009	−2720.018
(skew) NIG distribution	1363.116	−2718.231
(symmetric) GHYP distribution	1380.43	−2752.86
(skew) GHYP distribution	1380.531	−2751.062

In Table 3, it was found that the AIC values were all negative. Negative AIC values are due to large maximum log-likelihood values and as previously mentioned, the smaller the information criterion, the better the model is in comparison to other models. The results in Table 3 indicate that the symmetric GHYP distribution has the best fit as concluded by Chu et al. [13]. The GED distribution comparatively comes in a close second to the GHYP distributions whether symmetric or skew. The worst fit is obviously the Normal distribution as the Q-Q plot shows that our data is extremely leptokurtic. The skew variants of the distributions fitted to the data are shown to not do any better than their symmetric counterparts, as previous results also indicated a slightly skewed to the left distribution.

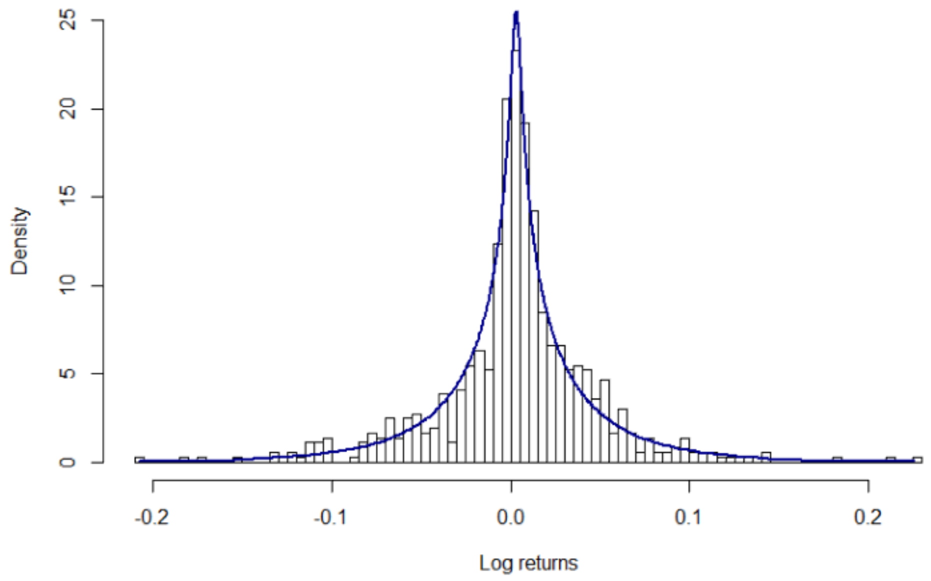


Figure 4: Histogram of log returns of BTC/USD exchange with a fitted GHYP curve.

Figure 4 displays a histogram of the BTC/USD exchange log returns and the fitted GHYP distribution curve. The

GHYP distribution (dark-blue) curve shows that it models the data very well. It would also suggest that when modelling the ARMA-GARCH to the data, that the GHYP distribution would most likely be the best conditional distribution.

3.6 ARMA-GARCH modelling

In this section we model the volatility of the daily USD/BTC log returns with several combinations of $ARMA\,(p,q)-GARCH\,(m,n)$ models. The ARMA model alone would also not suffice as it would not capture the data's full characteristics, and so in combination with the GARCH model used to model the volatility, a better fitted model can be obtained. Apart from the fact that GARCH models take into account the varying volatility (i.e. fluctuation of error variance) and the property of conditional heteroscedasticity (i.e. past experience), they also exhibit properties that are a good fit to heavier-tailed distributions. As previously mentioned the standard GARCH (sGARCH) assumes that the underlying distribution is normal. This proves to be inadequate in our case as a heavier tailed distribution would better suit our data as indicated from the results of our empirical analysis. In order to select the best fitted model, we make use of the four information criteria; Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Shibata Information Criterion (SIC) and Hannan-Quin Information Criterion (HQIC). They are defined as follows

$$AIC = \frac{-2\ln L\,(\Theta)}{n} + \frac{2K}{n}. \tag{41}$$

$$BIC = \frac{-2\ln L\,(\Theta)}{n} + \frac{K\ln(n)}{n}. \tag{42}$$

$$SIC = \frac{-2\ln L\,(\Theta)}{n} + \frac{2K\ln(\ln(n))}{n}. \tag{43}$$

$$HQIC = \frac{-2\ln L\,(\Theta)}{n} + \ln\left(\frac{n+2K}{n}\right). \tag{44}$$

where $\ln L\,(\Theta)$ represents the maximum log-likelihood, K being the number of parameters estimated using maximum likelihood estimation (MLE) and n is the total number of observations.

We fit the daily USD/BTC log returns with a combination of $ARMA\,(p,q) - sGARCH\,(m,n)$ under different underlying distribution. The parameters of the underlying distribution are estimated using MLE. The model fitting information criterion selection results are listed in the tables below.

Table 4: Model fitting results (AIC) under different conditional distributions, η_t.

	Normal	Student's t-distribution	GED	(skew) GED	GHYP	NIG	JSU
$ARMA(0,0)-GARCH(1,1)$	−3.8124	−3.9851	−4.0112	−4.0096	−4.0078	−4.0051	−3.9974
$ARMA(1,0)-GARCH(1,1)$	−3.8105	−3.9824	−4.0098	−4.0088	−4.0063	−4.0027	−3.9947
$ARMA(0,1)-GARCH(1,1)$	−3.8104	−3.9824	−4.0099	−4.0089	−4.0064	−4.0027	−3.9947
$ARMA(1,1)-GARCH(1,1)$	−3.8083	−3.9901	−4.0073	−4.0064	−4.0039	−4.0000	−3.9920
$ARMA(2,1)-GARCH(1,1)$	−3.8037	−3.9853	−4.0168	−4.0037	−4.0003	−3.9950	−3.9866
$ARMA(1,2)-GARCH(1,1)$	−3.8038	−3.9853	−4.0042	−4.0032	−4.0001	−3.9950	−3.9868
$ARMA(2,2)-GARCH(1,1)$	−3.8014	−3.9821	−4.0185	−4.0163	−4.0079	−4.0066	−3.9948

Table 5: Model fitting results (BIC) under different conditional distributions, η_t.

	Normal	Student's t-distribution	GED	(skew) GED	GHYP	NIG	JSU
$ARMA(0,0)-GARCH(1,1)$	−3.7872	−3.9537	−3.9798	−3.9718	−3.9638	−3.9673	−3.9596
$ARMA(1,0)-GARCH(1,1)$	−3.7791	−3.9446	−3.9721	−3.9648	−3.9559	−3.9586	−3.9507
$ARMA(0,1)-GARCH(1,1)$	−3.7790	−3.9446	−3.9721	−3.9649	−3.9560	−3.9586	−3.9507
$ARMA(1,1)-GARCH(1,1)$	−3.7705	−3.9461	−3.9633	−3.9560	−3.9473	−3.9496	−3.9417
$ARMA(2,1)-GARCH(1,1)$	−3.7597	−3.9350	−3.9665	−3.9471	−3.9374	−3.9384	−3.9300
$ARMA(1,2)-GARCH(1,1)$	−3.7598	−3.9350	−3.9539	−3.9466	−3.9372	−3.9384	−3.9302
$ARMA(2,2)-GARCH(1,1)$	−3.7510	−3.9255	−3.9619	−3.9533	−3.9387	−3.9437	−3.9318

Table 6: Model fitting results (SIC) under different conditional distributions, η_t.

	Normal	Student's t-distribution	GED	(skew) GED	GHYP	NIG	JSU
$ARMA(0,0)-GARCH(1,1)$	−3.8125	−3.9852	−4.0113	−4.0097	−4.0080	−4.0052	−3.9975
$ARMA(1,0)-GARCH(1,1)$	−3.8106	−3.9825	−4.0100	−4.0090	−4.0065	−4.0028	−3.9949
$ARMA(0,1)-GARCH(1,1)$	−3.8105	−3.9825	−4.0100	−4.0091	−4.0066	−4.0029	−3.9949
$ARMA(1,1)-GARCH(1,1)$	−3.8084	−3.9903	−4.0075	−4.0066	−4.0042	−4.0002	−3.9922
$ARMA(2,1)-GARCH(1,1)$	−3.8039	−3.9855	−4.0171	−4.0040	−4.0007	−3.9953	−3.9869
$ARMA(1,2)-GARCH(1,1)$	−3.8040	−3.9856	−4.0045	−4.0035	−4.0005	−3.9953	−3.9871
$ARMA(2,2)-GARCH(1,1)$	−3.8016	−3.9824	−4.0188	−4.0166	−4.0084	−4.0070	−3.9951

Table 7: Model fitting results ($HQIC$) under different conditional distributions, η_t.

	Normal	Student's t-distribution	GED	(skew) GED	GHYP	NIG	JSU
$ARMA(0,0)-GARCH(1,1)$	−3.8027	−3.9730	−3.9991	−3.9950	−3.9908	−3.9905	−3.9828
$ARMA(1,0)-GARCH(1,1)$	−3.7984	−3.9678	−3.9953	−3.9918	−3.9868	−3.9857	−3.9777
$ARMA(0,1)-GARCH(1,1)$	−3.7983	−3.9678	−3.9953	−3.9920	−3.9869	−3.9857	−3.9777
$ARMA(1,1)-GARCH(1,1)$	−3.7939	−3.9731	−3.9903	−3.9869	−3.9821	−3.9806	−3.9726
$ARMA(2,1)-GARCH(1,1)$	−3.7867	−3.9659	−3.9974	−3.9819	−3.9760	−3.9732	−3.9648
$ARMA(1,2)-GARCH(1,1)$	−3.7868	−3.9659	−3.9848	−3.9813	−3.9758	−3.9731	−3.9649
$ARMA(2,2)-GARCH(1,1)$	−3.7819	−3.9603	−3.9967	−3.9920	−3.9812	−3.9824	−3.9705

From the model fitting results in Tables 4 to 6, it was found that the information criterion were all negative. Negative information criterion are due to a large maximum log-likelihood and as previously mentioned, the smaller the information criterion, the better the model is in comparison to its counterparts. We have therefore, found that the best-fitted model with the smallest information criterion values was found to be an $ARMA(2,2)-GARCH(1,1)$ under a generalised error distribution (GED) whilst Tables 5 and 7 indicate $ARMA(0,0)-GARCH(1,1)$ under GED.

Model 1. Given the model's parameter estimates, the $ARMA(2,2)-GARCH(1,1)$ model with GED as the conditional distribution is defined as follows

$$\begin{aligned}\hat{Y_t} &= 0.002965 + 1.318506Y_{t-1} - 0.331822Y_{t-2} + \varepsilon_t - 1.376960\varepsilon_{t-1} + 0.408240\varepsilon_{t-2}.\\ \hat{\varepsilon_t} &= \hat{\sigma_t}\eta_t.\\ \hat{\sigma_t^2} &= 0.000025 + 0.195925\varepsilon_{t-1}^2 + 0.803073\sigma_{t-1}^2.\end{aligned}$$

where the conditional distribution is a $\eta_t \sim GED\,(0,1)$ (i.e. the mean and variance are standardized to be $\mu = 0$ and $\sigma^2 = 1$) with a shape parameter of $\hat{\nu} = 0.869724$ which captures the leptokurtic nature of the data.

We also observe that in the comparison of the information criterion values that the $ARMA(0,0)-GARCH(1,1)$, under the GED, was the better fitted model in other cases.

Model 2. The model is defined as follows

$$\begin{aligned}\hat{Y_t} &= 0.002565 + \varepsilon_t.\\ \hat{\varepsilon_t} &= \hat{\sigma_t}\eta_t.\\ \hat{\sigma_t^2} &= 0.000025 + 0.208038\varepsilon_{t-1}^2 + 0.790962\sigma_{t-1}^2.\end{aligned}$$

where the conditional distribution is a $\eta_t \sim GED\,(0,\,1,\,0.941244)$ (i.e. the shape parameter is estimated to be $\hat{\nu} = 0.941244$)which captures the leptokurtic nature of the data.

We also found that keeping all other variables constant that increasing the $GARCH\,(m,n)$ parameter (i.e. $m = n = 2$) did not produce a better model. We had also, however, previously anticipated that given any $ARMA\,(p,q) - sGARCH\,(m,n)$ we fit, that the underlying distribution would be the GHYP distribution. This did not prove to be completely true in our case, although it came in a close third to being chosen, whilst the skewed variant of the GED came in second. We did previously mention that in our empirical analysis that both the GED and GHYP distributions were close in AIC values and thus both models would be good fits to our data. This, therefore, indicates the reliability of our conclusions.

3.7 Forecasting

Given the two models estimated as the best fits to the USD/BTC daily log returns in the previous sections, we can gain better insights into their predictability by forecasting. The unconditional volatility forecast of 10 days ahead is based only on the parameter estimates of the models fitted. This is depicted in the figure below.

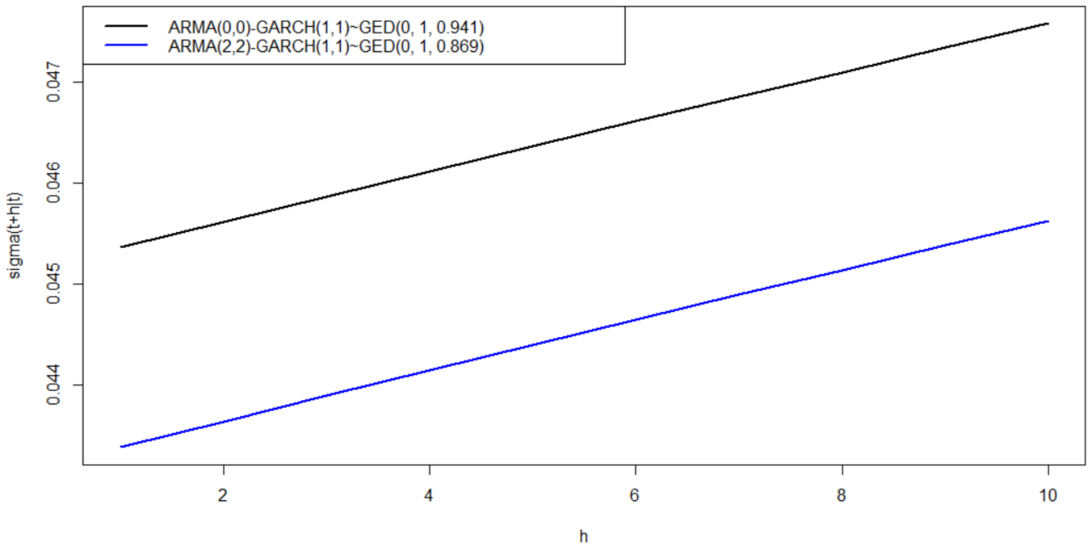
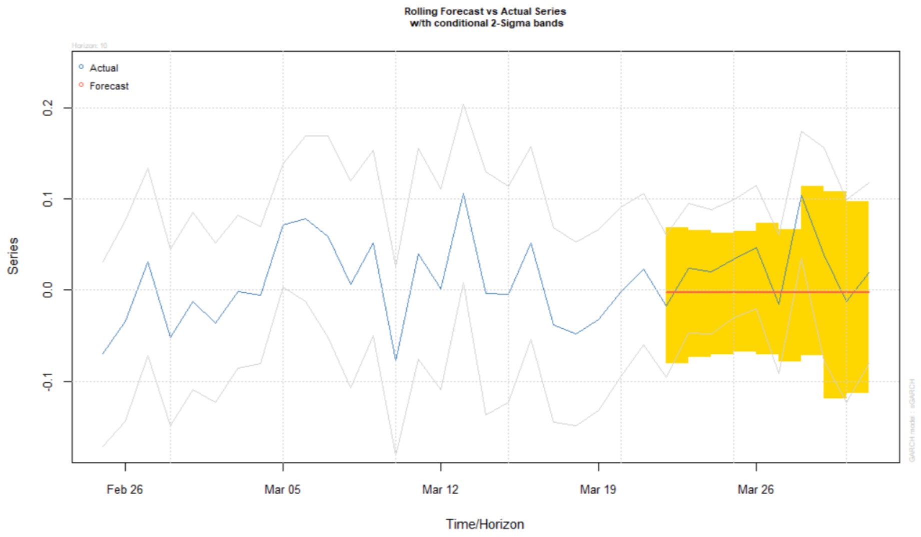


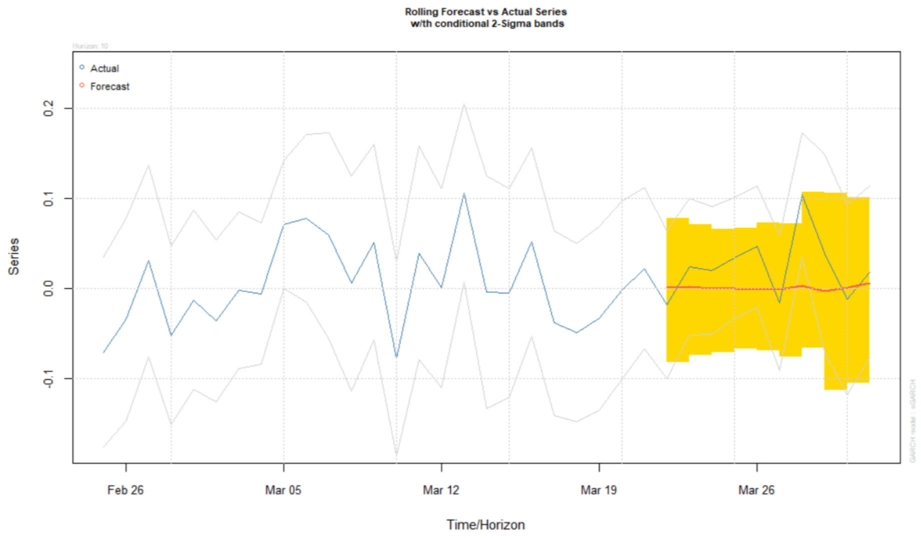
Figure 5: 10-day unconditional volatility forecasting of the competing models.

Figure 5 depicts both competing models' volatility forecasts; 10 days into the future. Both models indicate an increase in volatility over time expected in future. We note that the volatility forecasts of Model 2 increase at a slightly higher rate than that of Model 1. This may indicate that a greater shape leads to the volatility forecasted also increasing at a higher rate.

We then went on to analyse the rolling forecasts of the competing models. These rolling forecasts are conditional on the past information as 10 values were sampled out and then the remaining USD/BTC daily log returns were remodeled using the estimated models. In the following figures, we will display the corresponding 10-day rolling forecasted time series and volatility series for each competing model.



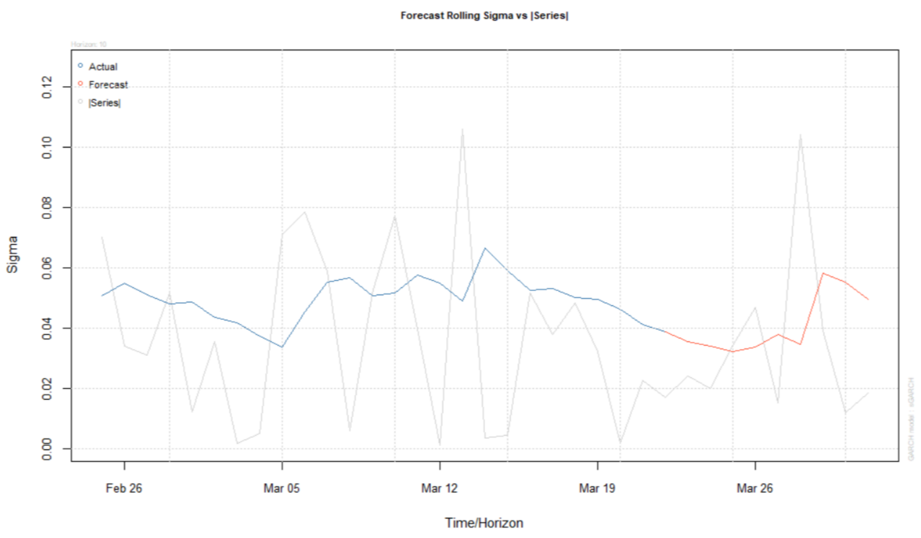
(*i*)



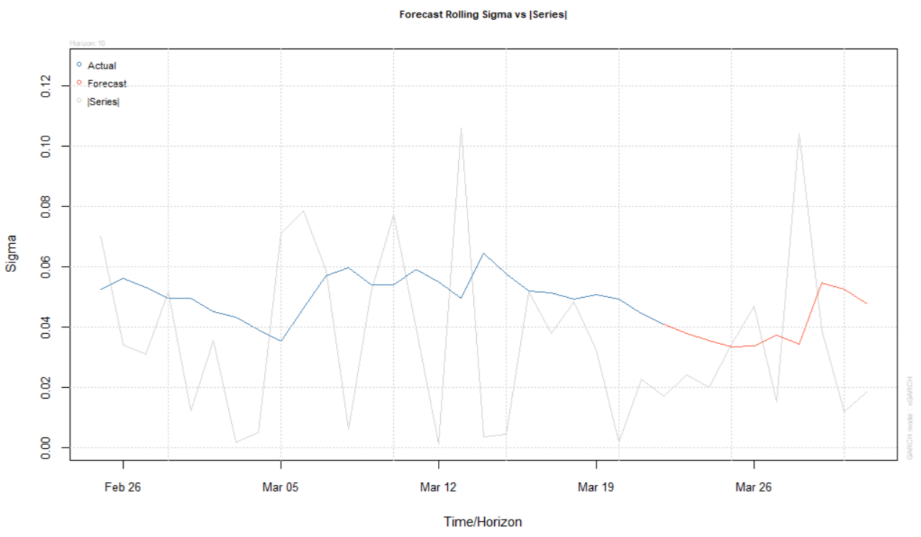
(*ii*)

Figure 6: 10-day rolling forecasted time series of the competing models: (*i*) Model 1 and (*ii*) Model 2.

In Figure 6, the rolling forecasted time series of length 10 are displayed for each estimated model. Although both model forecasts show mean reversion, Model 2's forecasts are slightly better than those of Model 1 . This is due to the fact that the forecasts of Model 1 seem to remain constant at zero, while Model 2's forecasts better follow the actual data's mean reversion characteristics with some variation like the data. Both models have an underlying $GED\,(0,\,1)$ with different shape parameters. Model 2 may be seen to perform better than Model 1 due to Model 2's shape parameter being slightly smaller than that of Model 1 which may indicate that the underlying distribution is slightly less heavy-tailed. The results above also indicate that a more intricate model apart from the standard GARCH would produce better forecasts. Analysis of Bitcoin returns was performed by Vo and Xu [57] in 2017 and they found that $ARMA(1,2) - fGARCH(2,2)/T - GARCH$ was the best estimated model of the USD/BTC hourly log returns.



(*i*)



(*ii*)

Figure 7: 10-day rolling forecasted volatility (sigma) series of the competing models: (*i*) Model 1 and (*ii*) Model 2.

In Figure 7, the rolling forecasted sigma series or standard deviation series of length 10 are displayed for each estimated model. There is not much of a difference between the sigma forecasts of Model 1 and Model 2. This may be due to the fact that the underlying distributions are both $GED\,(0,\,1)$ with slightly different shape parameters. Considering the similarities of both competing models' volatility forecasts, we can definitely say that they both estimate the sigma series with good precision than that of the time series in Figure 6. Although, they still show a lot of deviation from the observed series.

Note that all the figures and tables produced have been computed in R [14] and the relevant code can be found in Appendix B.

4 Linear Regression with ARMA-GARCH errors

In this chapter, we will introduce the linear regression models with non-normal errors, particularly $GARCH$ and $ARMA-GARCH$ errors. We will also develop and derive the theory of a linear regression model with $ARMA-GARCH$ errors.

4.1 Linear Regression with GARCH Errors

One of the most widely used statistical tools when evaluating and enhancing our understanding of the relationship of how one or more variables (predictors) affects another (response), regression analysis is used. It is simplistic, easy to work with and interpret, provided that none of the model's assumptions have been violated using Ordinary Least Square Estimation (OLSE). Econometricians and in particularly macroeconomists develop models to explain the effects of how factors such as inflation, international trade and/or national income affect the economy as a whole in order to aid the evaluation and further development of government economic policies. One particularly incumbent drawback of linear regression is such that when any of its assumptions are violated, for example in the case of multicollinearity, even though the results may still be reliable, the calculations of the individual predictors would be affected. Some of the assumptions required for OLSE in regression analysis include that the error terms should be uncorrelated over time. This is typically violated by time series regression giving rise to autocorrelation. Even though the estimated regression coefficients possibly remain unbiased, consequently they become inefficient and the corresponding standard errors are also probably erroneous. This results in all other statistics calculated inapplicable (e.g. t-statistic, confidence intervals and F-statistic). Another assumption of OLSE is that the error terms must be independent and identically, normally distributed with a mean of zero and a constant variance (i.e. homoscedasticity). If this assumption is violated, meaning heteroscedasticity is present, the regression coefficient estimated remains unbiased and consistent but lose efficiency. The standard errors again will be invalid which will lead to inaccurate precision of results.

Regression analysis is a powerful tool that allows us to study the effect of changes in the various factors on the predictors. If the observations of the dependent and independent variables are measured over time then a time series regression model can be built. We can then easily predict the future target values given the results of the statistical relationship of the variables.

Time series data can be described as a sequence of successive, equidistant measurements collected on the same observational unit over multiple time periods. For example, the price of Bitcoin observed at different time intervals (e.g. hourly, daily, weekly, monthly etc.) over a designated period of time. Time series analysis is the use of statistical methods to analyse and extract meaningful statistics and characteristics of the time series data. This typically involves modelling a variable which is dependent on its past values, and/or modelling the present and past values on other variables. In particular time series regression is commonly use to estimate the linear relationship of one or more time series to a a response time series variable. As mentioned previously, the regression coefficients are estimated through OLSE under certain assumptions. One necessary condition being that the variance of the errors (residuals) must be independent and constant; homoscedastic. This is typically not the case when it comes to financial data. Thus regression models have been studied and developed to accommodate the characteristics that observed data embody, such as heteroscedasticity.

Van den Bossche et al. [55] developed a regression model with ARIMA errors to investigate the impact of weather, laws and regulations, and economic conditions on the frequency and severity of accidents in Belgium. Their study revealed that the regression model with ARIMA errors gives an acceptable fit. It also showed that weather conditions and some policy regulations had significant influence on traffic safety. Although economic policy proved to be insignificant, the author did warn that the regression models with ARIMA errors could become quite complex and advises that the most parsimonious model should be used. Another advancement in this area of time series regression is that of linear regression models with GARCH errors. Hossain and Ghahramani [29] introduced and derived the asymptotic theory of shrinkage estimation of a linear regression model with GARCH errors. The regression model is defined as follows

Definition 4.1.1. Let Y_t be the t-th observation of the response variable is modelled as follows

$$Y_t = X_t^{'}\beta + \varepsilon_t, \qquad t = 1, 2, \ldots, T.$$

where $X_t^{'} = (X_{t1}, X_{t2}, \ldots, X_{tk})$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_k)^{`}$ are both $k \times 1$ vectors; of the predictors and unknown regression coefficients respectively. ε_t is the error term which follows a $GARCH\,(p, q)$ process such that

$$\varepsilon_t = \sigma_t \eta_t.$$

There exists nonnegative constants ,$\omega > 0$; $\alpha_i \geq 0$, $i = 1, 2, \ldots, p$ and $\gamma_j \geq 0$, $j = 1, 2, \ldots, q$ such that

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \gamma_j \sigma_{t-j}^2, \qquad \forall\, t \in \mathbb{Z}.$$

where η_t is a sequence of i.i.d. random variables such that $\eta_t \sim N\,(0, 1)$.

Our goal in this mini-dissertation is extend on the theory developed by looking into linear regression models with ARMA-GARCH errors.

4.2 Linear Regression with ARMA-GARCH Errors

First let us consider a regression model with a constant conditional variance such that

$$Var\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right) \quad = \quad \sigma^2.$$

Then it follows that the general form of the regression model of Y_t on the explanatory variables $X_t^{'}=\left(X_{t1},X_{t2},\ldots,X_{tk}\right)$ is given by

$$Y_t \quad = \quad f\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right)+\varepsilon_t \qquad \forall t.$$

where the error term, $\varepsilon_t\sim N\left(0,\,\sigma_\varepsilon^2\right)$ and $X_t^{'}=\left(X_{t1},X_{t2},\ldots,X_{tk}\right)$ are independent, f is the conditional function of Y_t given the explanatory variables $X_{t1},X_{t2},\ldots,X_{tk}$. As previously mentioned, in this case the conditional variance is constant and equivalent to the variance of the error term, σ_ε^2.

Next let us consider that our conditional variance of Y_t is no longer constant such that

$$Var\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right) \quad = \quad \sigma^2\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right).$$

Then the general regression model can be rewritten as

$$Y_t \quad = \quad f\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right)+\sigma\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right)\varepsilon_t \qquad \forall t.$$

where the error term, $\varepsilon_t\sim N\left(0,\,1\right)$ since it is conditional on $X_{t1},X_{t2},\ldots,X_{tk}$. The standard deviation $\sigma\left(Y_t|X_{t1},X_{t2},\ldots,X_{tk}\right)$ must be nonnegative and if the function $\sigma\left(\cdot\right)$ is linear, then its coefficients must be constrained in order to ensure nonnegativity. Nonlinear nonnegative functions are preferred as the constraints of linear functions are cumbersome to perform and execute. Models such as the GARCH model were developed to take into account the conditional variance and is broadly referred to as a variance function model.

Definition 4.2.1. Let Y_t be the t-th observation of the response variable is modelled as follows

$$Y_t \quad = \quad X_t^{'}\beta+\varepsilon_t, \qquad t=1,2,\ldots,T. \tag{45}$$

where $X_t^{'}=\left(X_{t1},X_{t2},\ldots,X_{tk}\right)$ and $\beta=\left(\beta_1,\beta_2,\ldots,\beta_k\right)^{`}$ are both $k\times 1$ vectors of the predictors and unknown regression coefficients respectively. ε_t is the error term which follows a $ARMA\left(p,q\right)-GARCH\left(m,n\right)$ process such that

$$\begin{aligned} \varepsilon_t &= c + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \ldots + \phi_p \varepsilon_{t-p} + \theta_1 \eta_{t-1} + \theta_2 \eta_{t-2} + \ldots + \theta_q \eta_{t-q} + \sigma_t \eta_t. \\ &= c + \sum_{i=1}^{p} \phi_i \varepsilon_{t-i} + \sum_{i=1}^{q} \theta_i \eta_{t-i} + \sigma_t \eta_t, \qquad \forall\, t \in \mathbb{Z}. \end{aligned} \tag{46}$$

where c could be any constant, $\phi_1, \phi_2, \ldots, \phi_p$ are the $AR(p)$ coefficients, $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients of the corresponding innovations (i.e. $MA(q)$ process); $\eta_t\ \forall\, t \in \mathbb{Z}$ is a sequence of i.i.d. random variables such that $\eta_t \sim N(0,1)$ (i.e. as defined in Definition 2.1.3.).

There also exists nonnegative constants ,$\omega > 0$ $\alpha_i \geq 0,\ i = 1, 2, \ldots, n$ and $\gamma_j \geq 0,\ j = 1, 2, \ldots, m$ associated with the $GARCH(m, n)$ process such that

$$\sigma_t^2 = \omega + \sum_{i=1}^{n} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{m} \gamma_j \sigma_{t-j}^2, \qquad \forall t \in \mathbb{Z}. \tag{47}$$

It should also be noted that ε_t can alternatively be expressed in the following manner

$$\begin{aligned} \left(1 - \sum_{i=1}^{p} \phi_i L^i\right) \varepsilon_t &= c + \left(\sigma_t + \sum_{i=1}^{q} \theta_i L^i\right) \eta_t, \qquad \forall\, t \in \mathbb{Z}. \\ \phi(L)\, \varepsilon_t &= c + \theta^\star(L)\, \eta_t. \end{aligned} \tag{48}$$

by dividing both sides of (48) by $\phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p = \sum_{i=1}^{p} \phi_i L^i$. We then have that ε_t rewritten in the following form

$$\begin{aligned} \varepsilon_t &= \frac{c}{\phi(L)} + \frac{\theta^\star(L)}{\phi(L)} \eta_t. \\ &= \mu + \Psi^\star(L)\, \varepsilon_t, \qquad \forall\, t. \end{aligned} \tag{49}$$

where μ is as defined in (21) and $\Psi^\star(L) = \frac{\theta^\star(L)}{\phi(L)}$.

It is common practice that we impose constraints on the parameters in the proposed model:

C1: As stated in (15), all the roots of $1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q = 0$ lie outside the unit circle.

C2: As stated in (16), all the roots of $1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p = 0$ lie outside the unit circle.

C3: $\alpha_i \geq 0,\ i = 1, 2, \ldots, p$ and $\gamma_j \geq 0,\ j = 1, 2, \ldots, q$.

C4: We also have $A(L) = \sum_{i=1}^{n} \alpha_i L^i$ and $J(L) = \sum_{j=1}^{m} \gamma_j L^j$ such that $A(L) + J(L) < 1$.

The first two constraints are in relation to the stationarity criterion and invertibility of the $ARMA(p, q)$ process. The third constraint is applied in order to guarantee that the σ_t^2, the conditional variance is positive always. The fourth constraint is usually applied for the finiteness of the unconditional variance of the GARCH innovations as done in previous GARCH modelling studies.

The dynamic and recursive nature of the conditional variance of the model , there are assumptions required for initializing the conditional variance and the squared residuals when $t \geq 1$. Hence the initialized values required are $\tilde{\varepsilon}_0^2,\, \tilde{\varepsilon}_1^2, \ldots,\, \tilde{\varepsilon}_{1-n}^2,\, \widetilde{\sigma}_0^2,\, \widetilde{\sigma}_1^2, \ldots,\, \widetilde{\sigma}_{1-m}^2$. For instance the initial values can be chosen as

$$\tilde{\varepsilon}_0^2 \;=\; \ldots = \tilde{\varepsilon}_{1-n}^2 = \widetilde{\sigma}_0^2 = \ldots = \widetilde{\sigma}_{1-m}^2 = \omega. \tag{50}$$

such that the vector of initialized values is given by $z_t = (\omega,\, \omega, \ldots,\, \omega)'$of the dimensions $(1+n+m)\times 1$ or the initial values can be as

$$\tilde{\varepsilon}_0^2 = \ldots = \tilde{\varepsilon}_{1-n}^2 = \widetilde{\sigma}_0^2 = \ldots = \widetilde{\sigma}_{1-m}^2. \tag{51}$$

such that the vector of initialised values is given by $z_t = \left(1,\, \tilde{\varepsilon}_0^2,\, \tilde{\varepsilon}_0^2 \ldots,\, \tilde{\varepsilon}_0^2\right)'$ of the dimensions $(1+n+m)\times 1$. This will be pertinent in the derivations of the information matrix.

It can proved that for a regression model with $GARCH$ errors, the conditional distribution of the errors is $\varepsilon_t|\Omega_{t-1} \sim N\left(0,\, \sigma_t^2\right)$ where Ω represents the information set, inclusive of all relevant information up to time period t. This is not necessarily the case for a regression model with $ARMA-GARCH$ errors. Provided that ε_t is defined as in (46) it follows that

$$\begin{aligned} E\left[\varepsilon_t|\Omega_{t-1}\right] &= E\left[c+\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+\sum_{i=1}^{q}\theta_i\eta_{t-i}+\sigma_t\eta_t\right] \\ &= E\left[c\right]+E\left[\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]+E\left[\sum_{i=1}^{q}\theta_i\eta_{t-i}\right]+E\left[\sigma_t\eta_t\right] \\ &= c+\sum_{i=1}^{p}\left(\phi_i E\left[\varepsilon_{t-i}\right]\right)+\sum_{i=1}^{q}\left(\theta_i E\left[\eta_{t-i}\right]\right)+E\left[\sigma_t\right]E\left[\eta_t\right] \end{aligned}$$

It is important to note that since η_t is a sequence of i.i.d. random variables such that $\eta_t \sim N(0,1)\ \forall\, t$; $E\left[\eta_t\right]=0$ and $Var\left[\eta_t\right]=1$. It is notable that η_t and σ_t are independent from one another. Also since the error term can also be written as $\varepsilon_t = Y_t - X_t^{'}\beta$, we can assume that the unconditional expected value of ε_t for all t are the same. Therefore

$$\begin{aligned} E\left[\varepsilon_t\right] &= E\left[\varepsilon_{t-1}\right]=E\left[\varepsilon_{t-2}\right]=\ldots=E\left[\varepsilon_{t-p}\right] \\ &= \hat{\mu} \\ &= \frac{1}{T}\sum_{t=1}^{T}Y_t. \end{aligned}$$

It then follows that

$$\begin{aligned} E\left[\varepsilon_t|\Omega_{t-1}\right] &= c+\sum_{i=1}^{p}\left(\phi_i\mu\right)+\sum_{i=1}^{q}\left(\theta_i\cdot 0\right)+E\left[\sigma_t\right]\cdot 0. \\ \mu &= c+\mu\sum_{i=1}^{p}\phi_i. \\ c &= \mu\left(1-\sum_{i=1}^{p}\phi_i\right). \end{aligned}$$

Then by dividing both sides by $1-\sum_{i=1}^{p}\phi_i$, we get that

$$\hat{\mu} = \frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}. \tag{52}$$

provided that

$$\sum_{j=0}^{\infty}|\Psi_j|<\infty.$$

and recall that stationarity depends on the roots of

$$\phi\left(z\right) = 1-\phi_1 z-\phi_2 z^2-\ldots-\phi_p z^p=0.$$

also lie outside the unit circle as stated in (16).

It is also now imperative that we determine the conditional variance of the $ARMA-GARCH$ errors. The variance will be of the following form

$$Var\left[\varepsilon_t|\Omega_{t-1}\right] = E\left[\varepsilon_t^2|\Omega_{t-1}\right]-\left(E\left[\varepsilon_t|\Omega_{t-1}\right]\right)^2.$$

where

$$E\left[\varepsilon_t|\Omega_{t-1}\right] = \frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}=\hat{\mu}.$$

as proved in (52). We then have to derive the other component of the variance, $E\left[\varepsilon_t^2|\Omega_{t-1}\right]$.

Let

$$\begin{aligned} \varepsilon_t^2 &= \left\{c+\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+\sum_{i=1}^{q}\theta_i\eta_{t-i}+\sigma_t\eta_t\right\}^2 \\ &= c^2+2c\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+2c\sum_{i=1}^{q}\theta_i\eta_{t-i}+2c\sigma_t\eta_t+\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2+2\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\} \\ &\quad +2\sigma_t\eta_t\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2+2\sigma_t\eta_t\sum_{i=1}^{q}\theta_i\eta_{t-i}+\sigma_t^2\eta_t^2. \end{aligned}$$

It then follows that

$$\begin{aligned}
E\left[\varepsilon_t^2|\Omega_{t-1}\right] &= E\left[\left\{c+\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+\sum_{i=1}^{q}\theta_i\eta_{t-i}+\sigma_t\eta_t\right\}^2\right] \\
&= E\left[\begin{array}{c} c^2+2c\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+2c\sum_{i=1}^{q}\theta_i\eta_{t-i}+2c\sigma_t\eta_t+\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2+2\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\} \\ +2\sigma_t\eta_t\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}+\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2+2\sigma_t\eta_t\sum_{i=1}^{q}\theta_i\eta_{t-i}+\sigma_t^2\eta_t^2 \end{array}\right] \\
&= E\left[c^2\right]+E\left[2c\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]+E\left[2c\sum_{i=1}^{q}\theta_i\eta_{t-i}\right]+E\left[2c\sigma_t\eta_t\right]+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] \\
&\quad +E\left[2\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]+E\left[2\sigma_t\eta_t\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]+E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right] \\
&\quad +E\left[2\sigma_t\eta_t\sum_{i=1}^{q}\theta_i\eta_{t-i}\right]+E\left[\sigma_t^2\eta_t^2\right] \\
&= c^2+2c\sum_{i=1}^{p}\left(\phi_iE\left[\varepsilon_{t-i}\right]\right)+2c\sum_{i=1}^{q}\left(\theta_iE\left[\eta_{t-i}\right]\right)+2cE\left[\sigma_t\right]E\left[\eta_t\right]+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] \\
&\quad +2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]+2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right] \\
&\quad +2E\left[\sigma_t\right]E\left[\eta_t\right]\sum_{i=1}^{q}\left(\theta_iE\left[\eta_{t-i}\right]\right)+E\left[\sigma_t^2\right]E\left[\eta_t^2\right].
\end{aligned}$$

As previously mentioned since η_t is a sequence of i.i.d. random variables such that $\eta_t \sim N(0,1)\ \forall\, t$, it also independent of σ_t. It should also be noted that $Var\left(\eta_t\right)=1=E\left[\eta_t^2\right]$ since $E\left[\eta_t\right]=0$. Recall that $E\left[\varepsilon_t\right]=\hat{\mu}\,\forall\, t$ as derived in (52). It is also important to note that $E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right]$ reduces to $\sum_{i=1}^{q}\theta_i^2$. For example when $q=3$,

$$\begin{aligned}
E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right] &= E\left[\left\{\theta_1\eta_{t-1}+\theta_2\eta_{t-2}+\theta_3\eta_{t-3}\right\}^2\right] \\
&= E\left[\theta_1^2\eta_{t-1}^2+\theta_2^2\eta_{t-2}^2+\theta_3^2\eta_{t-3}^2+\theta_1\theta_2\eta_{t-1}\eta_{t-2}+\theta_1\theta_3\eta_{t-1}\eta_{t-3}+\theta_2\theta_3\eta_{t-2}\eta_{t-3}\right] \\
&= E\left[\theta_1^2\eta_{t-1}^2\right]+E\left[\theta_2^2\eta_{t-2}^2\right]+E\left[\theta_3^2\eta_{t-3}^2\right]+E\left[\theta_1\theta_2\eta_{t-1}\eta_{t-2}\right] \\
&\quad +E\left[\theta_1\theta_3\eta_{t-1}\eta_{t-3}\right]+E\left[\theta_2\theta_3\eta_{t-2}\eta_{t-3}\right] \\
&= \theta_1^2E\left[\eta_{t-1}^2\right]+\theta_2^2E\left[\eta_{t-2}^2\right]+\theta_3^2E\left[\eta_{t-3}^2\right]+\theta_1\theta_2E\left[\eta_{t-1}\right]E\left[\eta_{t-2}\right] \\
&\quad +\theta_1\theta_3E\left[\eta_{t-1}\right]E\left[\eta_{t-3}\right]+\theta_2\theta_3E\left[\eta_{t-2}\right]E\left[\eta_{t-3}\right] \\
&= \left(\theta_1^2\cdot 1\right)+\left(\theta_2^2\cdot 1\right)+\left(\theta_3^2\cdot 1\right)+\left(\theta_1\theta_2\cdot 0\cdot 0\right)+\left(\theta_1\theta_3\cdot 0\cdot 0\right)+\left(\theta_2\theta_3\cdot 0\cdot 0\right) \\
&= \theta_1^2+\theta_2^2+\theta_3^2 \\
&= \sum_{i=1}^{3}\theta_i^2
\end{aligned}$$

It then follows that

$$
\begin{aligned}
E\left[\varepsilon_t^2|\Omega_{t-1}\right] &= c^2+2c\sum_{i=1}^{p}(\phi_i\mu)+2c\sum_{i=1}^{q}(\theta_i\cdot 0)+2cE\left[\sigma_t\right]\cdot 0+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] \\
&\quad +2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]+2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right] \\
&\quad +2E\left[\sigma_t\right]\cdot 0\cdot\sum_{i=1}^{q}(\theta_i\cdot 0)+E\left[\sigma_t^2\right]\cdot 1 \\
&\quad c^2+2c\mu\sum_{i=1}^{p}\phi_i+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\
&\quad +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+E\left[\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}^2\right]+E\left[\sigma_t^2\right] \\
&= c^2+2c\mu\sum_{i=1}^{p}\phi_i+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\
&\quad +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^{q}\theta_i^2+E\left[\sigma_t^2\right].
\end{aligned}
$$

Therefore the conditional variance of the $ARMA-GARCH$ errors is given as follows

$$
\begin{aligned}
Var\left[\varepsilon_t|\Omega_{t-1}\right] &= h_t^2 \\
&= E\left[\varepsilon_t^2|\Omega_{t-1}\right]-\left(E\left[\varepsilon_t|\Omega_{t-1}\right]\right)^2 \\
&= c^2+2c\mu\sum_{i=1}^{p}\phi_i+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\
&\quad +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^{q}\theta_i^2+E\left[\sigma_t^2\right]-\mu^2. \qquad (53)
\end{aligned}
$$

where μ is defined as in (52).

It was previously defined in Hossain and Ghahramani's [29] article that for a linear regression model with $GARCH$ errors that $\beta=(\beta_1,\beta_2,\ldots,\beta_k)'$ is a $(k\times 1)$ vector of unknown regression coefficients, $\delta'=(\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m)$ is the $(1+m+n)\times 1$ vector of unknown variance parameters. Then it was defined that $\xi=(\beta',\,\delta')$ was the $(k+1+m+n)\times 1$ vector of all unknown parameters such that $\xi\in\Xi$ and Ξ was defined as a compact subset of the Euclidean space. It was also noted that the GARCH errors could be reduced to ARCH errors provided that $q=0$ for $j=1,\,2,\ldots,\,q$. Now in the case of a linear regression model with $ARMA-GARCH$ errors, we also define the vector of unknown regression coefficients as $\beta=(\beta_1,\beta_2,\ldots,\beta_k)'$ as was done by Hossain and Ghahramani's [29] although the vector of unknown variance parameters differs to include the $ARMA$ component. Thus we define our $(2+p+q+m+n)\times 1$ vector of unknown variance parameters as $\delta'=(\delta_1',\,\delta_2')$ such that $\delta_1'=(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q):(1+p+q)\times 1$ and $\delta_2'=(\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m):(1+m+n)\times 1$. Then we may also extend the definition of the new vector of all unknown parameters which is $\xi=(\beta',\,\delta')$ of dimensions $(k+2+p+q+m+n)\times 1$ such that $\xi \in \Xi$ and Ξ was defined as a compact subset of the Euclidean space.

Let $\xi_0 = (\beta_{01},\, \beta_{02}, \ldots, \beta_{0k},\, c_0,\, \phi_{01},\, \phi_{02}, \ldots,\, \phi_{0p}\, \theta_{01},\, \theta_{02}, \ldots,\, \theta_{0q},\, \omega_0,\, \alpha_{01},\, \alpha_{02}, \ldots,\, \alpha_{0n}\, \gamma_{01},\, \gamma_{02}, \ldots,\, \gamma_{0m})'$ be the vector of all the unknown but true parameter values. Conditioning on initial values

$\varepsilon_0^2,\, \varepsilon_1^2, \ldots,\, \varepsilon_{1-n}^2\, \sigma_0^2,\, \sigma_1^2, \ldots,\, \sigma_{1-m}^2$, the Gaussian quasi-likelihood is given by

$$\mathcal{L}_{\mathcal{T}}(\xi) \quad = \quad \frac{1}{T}\sum_{t=1}^{T} \ell_t(\xi)\,. \tag{54}$$

Provided that the probability density function is of the form

$$f\left(x|\mu,\, \sigma^2\right) \quad = \quad \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

where $x,\, \mu\, \in \mathbb{R}$ and $\sigma^2\, \in \mathbb{R}^{\dotplus}$ such that $\mu,\, \sigma^2$ are the mean and variance respectively of the random variable X. Now consider $x = \varepsilon_t,\ \mu = E\left[\varepsilon_t|\Omega_{t-1}\right]$ as derived previously and $\sigma^2 = h_t^2 = Var\left[\varepsilon_t|\Omega_{t-1}\right]$ also derived previously. The new probability density function then appears as follows

$$f\left(\varepsilon_t|\mu,\, h_t^2\right) = \frac{1}{\sqrt{2\pi h_t^2}} exp\left\{-\frac{(\varepsilon_t-\mu)^2}{2h_t^2}\right\}.$$

The likelihood function of the model is derived as follows

$$\begin{aligned}
\ell_t(\xi) \quad &= \quad f\left(\varepsilon_1, \ldots, \varepsilon_t|\mu,\, h_t^2\right) \\
&= \quad f\left(\varepsilon_T|\Omega_{T-1}\right) \times f\left(\varepsilon_{T-1}|\Omega_{T-2}\right) \times \ldots \times f\left(\varepsilon_{s+1}|\Omega_s\right) \times f\left(\varepsilon_1, \ldots, \varepsilon_s|\mu,\, h_t^2\right) \\
& \quad \prod_{t=s+1}^{T} \frac{1}{\sqrt{2\pi\sigma_t^2,}} exp\left\{-\frac{(\varepsilon_t)^2}{2\sigma_t^2,}\right\} \times f\left(\varepsilon_1, \ldots, \varepsilon_s|\mu,\, h_t^2\right).
\end{aligned} \tag{55}$$

where s is the maximum order of the $ARMA(p,q) - GARCH\,(m,n)$ process $(s = max\,(p,\, q,\, m,\, n))$ and provided that

$$\varepsilon_t \quad = \quad \begin{cases} \varepsilon_0, & t = 0 \\ Y_t - X_t'\beta - \sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}'\beta\right) - \eta_t - \sum_{i=1}^{q}\theta_i\eta_{t-i}, & t = 1, 2, \ldots, T. \end{cases}$$

Then it follows that the log- likelihood function is given by

$$\begin{aligned}
\mathcal{L}_{\mathcal{T}}(\xi) &= log\left[\ell_t(\xi)\right] \\
&= \log\left(\prod_{t=s+1}^{T}\frac{1}{\sqrt{2\pi h_t^2}}exp\left\{-\frac{(\varepsilon_t-\mu)^2}{2h_t^2}\right\}\right) \\
&= \sum_{t=s+1}^{T} log\left[\frac{1}{\sqrt{2\pi h_t^2}}exp\left\{-\frac{(\varepsilon_t-\mu)^2}{2h_t^2}\right\}\right] \\
&= \sum_{t=s+1}^{T}\left[\log(2\pi)^{-\frac{1}{2}}+\log\left(h_t^2\right)^{-\frac{1}{2}}-\frac{1}{2}\frac{(\varepsilon_t-\mu)^2}{h_t^2}\right] \\
&= \sum_{t=s+1}^{T}\left[-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log\left(h_t^2\right)-\frac{1}{2}\frac{\left(\varepsilon_t^2-2\varepsilon_t\mu+\mu^2\right)}{h_t^2}\right] \\
&= -\frac{T}{2}\log(2\pi)-\frac{1}{2}\sum_{t=s+1}^{T}\log\left(h_t^2\right)-\frac{1}{2}\sum_{t=s+1}^{T}\frac{\varepsilon_t^2}{h_t^2}+\sum_{t=s+1}^{T}\frac{\varepsilon_t\mu}{h_t^2}-\frac{1}{2}\sum_{t=s+1}^{T}\frac{\mu^2}{h_t^2}. \qquad (56)
\end{aligned}$$

or the Gaussian quasi-likelihood (i.e. average log-likelihood) function can used which is given by

$$\begin{aligned}
\mathcal{L}_{\mathcal{T}}(\xi) &= \frac{1}{T}log\left[\ell_t(\xi)\right] \\
&= \frac{1}{T}\sum_{t=s+1}^{T}\left[\log(2\pi)^{-\frac{1}{2}}+\log\left(h_t^2\right)^{-\frac{1}{2}}-\frac{1}{2}\frac{(\varepsilon_t-\mu)^2}{h_t^2}\right] \\
&= -\frac{1}{2}\log(2\pi)-\frac{1}{T}\sum_{t=s+1}^{T}\frac{1}{2}\log\left(h_t^2\right)-\frac{1}{T}\sum_{t=s+1}^{T}\frac{1}{2}\frac{\varepsilon_t^2}{h_t^2}+\frac{1}{T}\sum_{t=s+1}^{T}\frac{\varepsilon_t\mu}{h_t^2}-\frac{1}{T}\sum_{t=s+1}^{T}\frac{1}{2}\frac{\mu^2}{h_t^2}. \qquad (57)
\end{aligned}$$

Therefore the quasi-maximum likelihood of ξ is defined as any measurable solution of $\hat{\xi_T}$ of

$$\begin{aligned}
\hat{\xi_T} &= \arg\max_{\xi\in\Xi}\mathcal{L}_{\mathcal{T}}(\xi) \\
&= \arg\min_{\xi\in\Xi}\bar{\mathcal{I}_{\mathcal{T}}}(\xi). \qquad (58)
\end{aligned}$$

where

$$\bar{\mathcal{I}_{\mathcal{T}}}(\xi) = \begin{bmatrix}\mathcal{I}_{\delta\delta} & 0 \\ 0 & \mathcal{I}_{\beta\beta}\end{bmatrix} \qquad (59)$$

is the information matrix. The variance-covariance matrix is an inverse of the information matrix. The components of the information matrix in (59) represent the second order derivatives of the parameters of $\xi = (\beta',\ \delta')$. The derivations of $\mathcal{I}_{\delta\delta}$ and $\mathcal{I}_{\beta\beta}$ can be found in the Appendix A. The off-diagonal blocks of the information matrix are $\mathcal{I}_{\delta\beta} = \mathcal{I}_{\beta\delta} = 0$.

5 Shrinkage Estimation

In this chapter we will introduce the idea of unrestricted, restricted and linear shrinkage estimators for the regression coefficients when the error terms follow an $ARMA-GARCH$ model. Shrinkage estimation is a widely studied topic in statistics and econometrics. In the books by Saleh [47] and Saleh et al. [49] [48], the insights of the preliminary test and related shrinkage estimation techniques are expanded upon and discussed. The authors provides a clear and practical introduction to shrinkage estimation in regression modelling.

As a prelude to shrinkage estimation, as in Saleh [47], the preliminary test approach of estimation which was first proposed by Bancroft [4] plays a systematic role in estimation. Preliminary test estimators are typically useful when one has to perform statistical inference with some uncertain prior information. We will also discuss the theory with regards to shrinkage estimation with respect to regression modelling and develop the relevant test statistic. We will then check validity of the model found in the previous chapter by way of application of the theory developed in this chapter on a real data example; specifically on Bitcoin price data, Google Trends [53] data on searches with regards to the term “bitcoin” and associated terms, and the S&P 500 index data.

5.1 Shrinkage Estimation for Multiple Regression Models

It is well-known that the use of prior information in the estimation of a statistical distribution’s parameters leads to improved results. Estimators that make use of prior information are referred to as restricted (R) estimators. Conversely, we refer to estimators without prior information as unrestricted (UR) estimators. So generally, restricted estimators often perform better than their unrestricted counterparts. In the event that the prior information is uncertain, preliminary test (PT) estimators are used. This method of estimation eliminates problematic parameters that arise from the prior information (i.e. constraints) incorporated into the model that are uncertain. The preliminary test will then determine whether the restricted or unrestricted model is then chosen based on the validity of the uncertain information leading to better results. It is advisable to make use of this prior information in the estimation procedure, especially when information based on the sample data may be limited. These prior information may be

1. A fact known from theoretical or experimental considerations,

2. A hypothesis that may have to be tested or,

3. An artificially imposed condition to reduce or eliminate redundancy in the description of the model (Rao and Debasis [43]).

This means that given these two extremes, a concession has to be made. The model in which no prior information is integrated into the final chosen model refers to the unrestricted model (i.e no restrictions imposed) whereas the restricted model is the instance where some or all prior information is incorporated from the final chosen model. For example, in a multiple regression model,

Consider the following simple multiple regression model

$$\begin{aligned} Y_t &= X_t'\beta + \varepsilon_t \\ &= \beta_0 + \beta_1 X_{t1} + \beta_2 X_{t2} + \beta_3 X_{t3} + \beta_4 X_{t4} + \beta_5 X_{t5} + \varepsilon_t, \qquad t = 1, 2, \ldots, T. \end{aligned} \tag{60}$$

where Y_t is t-th observation of the response (i.e dependent) variable, $\beta_j \,\forall\, j$ are the unknown regression coefficients estimated for the corresponding predictor (i.e independent) variables, X_t. Therefore X_{tj} is then the t-th observation of the j-th predictor and ε_t is the error term (i.e noise) or often referred to as the residual with a distribution as defined in Definition 2.1.3. The model defined in (60) is the unrestricted model.

Suppose we wish to test the following set of hypotheses

$$\begin{cases} H_0 & : \quad \beta_j = 0, \quad \forall\, j \\ H_A & : \quad \textit{at least one } \beta_j \textit{ is not zero.} \end{cases}$$

We are testing the hypothesis that all the regressors are insignificant versus the alternative that some or all are significantly different from zero. The restricted model in this case may deem some or all of the predictors insignificant, therefore the coefficients are set to zero and excluded from the final model. For example, the restricted model may be defined as follows

$$Y_{tR} = \hat{Y}, \qquad \forall\, t. \tag{61}$$

which indicates that all the predictors are insignificant. The results of the hypothesis test (i.e preliminary test) will then determine whether the unrestricted model is chosen as the final model.

The comparison of the fit of the unrestricted and restricted model will lead to an inference being made. Considering the fact that when estimating the regression coefficients, we aim to minimize the residual sum of squares. So in turn it is said the unrestricted model will fit at least as well as the restricted model since the unrestricted model is such that it can always choose a combination of the coefficients that the restricted model can.

Given the multiple regression model in (60), the UR estimator of β under maximum likelihood estimation (MLE) or least squares estimation (LSE) is defined as

$$\tilde{\beta} \quad = \quad \left(X'X\right)^{-1} X'Y. \tag{62}$$

where Y is a $(T \times 1)$ vector of the observed response (i.e dependent) variable, $\widetilde{\beta_{UR}}$ is a $(k \times 1)$ vector of the unrestricted and unknown regression coefficients that will be estimated such that they minimize the squared sum of residuals under LSE and maximize the likelihood function under MLE. X is referred as the design matrix of predictor (i.e. independent) variables; it has a full column rank $k(k < T)$ with dimensions $(T \times k)$. The URE of β is also distributed as follows

$$\tilde{\beta}_{UR} \quad \sim \quad N_k\left(\beta, \sigma^2 \left(X'X\right)^{-1}\right). \tag{63}$$

with the unrestricted and unbiased estimator of the variance σ^2 being given by

$$s_{\varepsilon}^2 \quad = \quad \frac{1}{T-k}\left(Y - X\tilde{\beta}\right)'\left(Y - X\tilde{\beta}\right). \tag{64}$$

Consider again the multiple regression model in (60), we then have that the RE, $\widehat{\beta}$ under maximum likelihood estimation (MLE) or least squares estimation (LSE) is defined as

$$\begin{aligned} \hat{\beta}_R &= R\beta \\ &= \tilde{\beta} - \left(X'X\right)^{-1} R'\left(R\left(X'X\right)^{-1} R'\right)^{-1}\left(R\tilde{\beta} - r\right). \end{aligned} \tag{65}$$

where we assumed that the restriction is

$$R\beta \quad = \quad r. \tag{66}$$

where R is a known and defined matrix with dimensions $(l \times k)$ of rank l $(l < k)$ and r being a pre-specified vector of dimensions $(l \times 1)$. Note that l represents the number of restrictions placed on the model.
As discussed previously, based on the restriction in (66), it will be determined whether the PTE will take on the unrestricted or restricted regression estimator. Given the multiple regression model in (60), the PTE of β is defined as follows

$$\hat{\beta}_{PT} \quad = \quad \tilde{\beta}_{UR} - \left(\tilde{\beta}_{UR} - \hat{\beta}_R\right) I_A\left(G\right). \tag{67}$$

where $I_A(G)$ is the indicator function of the set A such that

$$I_A(G) = \begin{cases} 1 & if\ G \in A. \\ 0 & if\ G \notin A. \end{cases}$$

This immediately follows that the PTE indicator function will be defined as follows

$$I_{UR}(\mathcal{F}) = I(\mathcal{L}_{\mathcal{T}} \leq \mathcal{F}(\alpha)) = \begin{cases} 1 & if\ H_0\ not\ rejected. \\ 0 & otherwise. \end{cases} \tag{68}$$

and

$$I_R(\mathcal{F}) = I(\mathcal{L}_{\mathcal{T}} > \mathcal{F}(\alpha)) = \begin{cases} 1 & if\ H_0\ rejected. \\ 0 & otherwise. \end{cases} \tag{69}$$

with the likelihood ratio test statistic denoted by $\mathcal{L}_{\mathcal{T}}$ defined by

$$\mathcal{L}_{\mathcal{T}} = \frac{\left(R\tilde{\beta} - r\right)\left(R\left(X'X\right)^{-1}R'\right)^{-1}\left(R\tilde{\beta} - r\right)}{l s_{\varepsilon}^2}. \tag{70}$$

based on the testing H_0 against H_A where $\mathcal{F}(\alpha)$ is the critical values of the $F-$distribution with degrees of freedom l and $(T-k)$, and a level of significance of α; described in Saleh [47].

The likelihood ratio test statistic under the alternative hypothesis has a noncentral $F-$distribution with degrees of freedom l and $(T-k)$, and a noncentrally parameter $\frac{\Delta^2}{2}$, where

$$\frac{\Delta^2}{2} = \frac{(R\beta - r)\left(R\left(X'X\right)^{-1}R'\right)^{-1}(R\beta - r)}{2\sigma^2}. \tag{71}$$

Lastly it should be noted that the UR and R estimator distributions are independent of the distribution of

$$\frac{(T-k)\, s_{\varepsilon}^2}{\sigma^2}.$$

5.2 Shrinkage estimation of linear regression with ARMA-GARCH errors

Before we discuss the shrinkage estimation with respect to linear regression models with $ARMA-GARCH$ errors, we will give an overview of shrinkage estimation. As previously described, preliminary test estimation has some ridgidity in that the restricted or unrestricted model is the chosen based on the outcome of the preliminary test.

Saleh [47] who provides extensive literature on this topic describes the nature of optimal PTE as being heavily dependent on the level of significance of the test. The optimal PTE only provides two choices for the estimator, namely the restricted and unrestricted estimator based on the outcome of the preliminary test. Saleh [47] suggests that in order to deal with these issues surrounding the estimator, shrinkage estimators (SE's) may be the alternative. The SE's will shrink toward a targeted prior value of the parameter under consideration and provide an estimator that incorporates both the extremes; restricted and unrestricted estimators. Essentially adjust the unrestricted estimator (for the full model) by the amount of the difference between unrestricted and restricted estimators scaled by the adjusted test-statistics for the uncertain prior information. The interpolated values depend on the value of the test statistic and not on the value of the test result.

Suppose $p = 2$. In our estimation of the optimal parameter of the regression model in (45), the regression coefficient vector is given by $\hat{\beta}_2$

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} \quad : \quad (2 \times 1) \tag{72}$$

where $\hat{\beta}_1$ and $\hat{\beta}_2$ represent the sub-parameters corresponding to the two predictor variables in $X_t^{'} = (X_{t1}, X_{t2})$. The regression model can then be rewritten as

$$\begin{aligned} Y_t &= X_t^{'}\beta + \varepsilon_t \\ &= \beta_1 X_{t1} + \beta_2 X_{t2} + \varepsilon_t, \qquad \forall\, t. \end{aligned} \tag{73}$$

The restricted model estimator of $\hat{\beta}$ can be obtained by maximizing the log-likelihood function where $\beta_2 = 0$ has been imposed as the constraint. The preliminary hypotheses are given by

$$\begin{cases} H_0 : & \beta_2 = 0 \\ H_A : & \beta_2 \neq 0 \end{cases} \tag{74}$$

such that the results of the test will be lead to the PT estimator being

$$\hat{\beta}_{PT} = \begin{cases} \tilde{\beta}_{UR} & reject\ H_0. \\ \hat{\beta}_R & do\ not\ reject\ H_0. \end{cases} \tag{75}$$

This indicates that if the restriction placed on the model is true (i.e. the null hypothesis is not rejected). Then the estimator of $\hat{\beta}$ chosen will be

$$\hat{\beta}_R = \begin{pmatrix} \hat{\beta}_1 \\ 0 \end{pmatrix}. \tag{76}$$

as it will presumably perform better than

$$\tilde{\beta}_{UR} = \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \end{pmatrix} \tag{77}$$

Although, ordinarily the restricted model tends to be biased, inefficient and inconsistent. In order to test the hypothesis, we consider a test statistic for the null hypothesis given as follows

$$\begin{aligned} \mathcal{D}_{\mathcal{T}} &= 2\left[\mathcal{L}\left(\tilde{\beta}_{UR}\,;Y_1,Y_2,\ldots,Y_{\mathcal{T}}\right) - \mathcal{L}\left(\hat{\beta}_R\,;Y_1,Y_2,\ldots,Y_{\mathcal{T}}\right)\right] \\ &= \mathcal{T}\tilde{\beta}_2^{\top}\mathcal{I}^{*}\tilde{\beta}_2. \end{aligned} \tag{78}$$

where $\mathcal{T}$ is the total number of observations, $\tilde{\beta}_2$ is the restriction placed on the model and

$$\mathcal{I}^{*} = \mathcal{I}_{22} - \frac{\mathcal{I}_{12} \times \mathcal{I}_{21}}{\mathcal{I}_{11}} = \mathcal{I}_{22} - \mathcal{I}_{21}\mathcal{I}_{11}^{-1}\mathcal{I}_{12}.$$

provided by the components of the variance-covariance matrix given by

$$\mathcal{I}\left(\tilde{\beta}_{UR}\right) = \begin{pmatrix} \mathcal{I}_{11} & \mathcal{I}_{12} \\ \mathcal{I}_{21} & \mathcal{I}_{22} \end{pmatrix} \tag{79}$$

Since we are testing whether the null hypothesis of $\hat{\beta}_2 = 0$ is true, as $\mathcal{T} \to \infty$ the test statistic is asymptotically chi-square distributed. We will then compare the test statistic in (78) to a chi-square critical value with one degree of freedom (i.e. $\chi^2_{(1;\pi)}$) at $(100 \times \pi)\,\%$ level of significance.

Alternatively to the preliminary test estimators we will have the linear shrinkage estimator (SE), $\hat{\beta}_S$, which is a linear combination of the UR estimator, $\tilde{\beta}_{UR}$ and the R estimator, $\hat{\beta}_R$. Linear shrinkage estimation does not depend on a test statistic but is simply a weighted estimate of the RE and URE, as follows $\hat{\beta}_R$

$$\hat{\beta}_S\left(\pi\right) = \pi\tilde{\beta}_{UR} + \left(1-\pi\right)\hat{\beta}_R \tag{80}$$

where π determines the extent to which these estimates are amalgamated. The parameter π is referred to as the degree of confidence in the null hypothesis such that if $\pi = 0$ then there will be shrinkage and the restricted will subjugate the model, and if $\pi = 1$ then there will be no degree of confidence in the null hypothesis. The criterion for comparing the performance of the SE, $\beta_S\hat{}(\pi)$ is the mean squared error (MSE) defined as follows

$$MSE\left(\pi\right) = \frac{1}{\mathcal{T}}\sum_{i=1}^{\mathcal{T}}\left(\hat{\beta}_S\left(\pi\right) - \beta_0\right)^{\top}\left(\hat{\beta}_S\left(\pi\right) - \beta_0\right). \tag{81}$$

where β_0 is the true value of β, based on the outcome of the hypothesis test. It should be noted that, the optimal SE is the one that provides the minimum mean squared error (MSE).

5.3 ARMA-GARCH regression modelling application

In this section, we will then apply our shrinkage estimation techniques discussed in the previous chapter on a sample of Bitcoin data. Preceding the analysis on the real data set, we will run a simulation study. We will document our results and verify that our predicted model is correct.

5.3.1 Simulation study

In this section, we demonstrate a Monte Carlo simulation study of the theory developed in the previous chapter in order to assess the relative performance of the proposed estimators with respect to the full model estimator. We consider the following multiple linear regression model with $ARMA - GARCH$ errors:

$$Y_t = X_t'\beta + \varepsilon_t, \qquad t = 1, 2, \dots, T. \tag{82}$$

where $X_t = (X_{t1}, X_{t2})'$ and $\beta = (\beta_1, \beta_2)'$ are both 2×1 vectors of the predictors and unknown regression coefficients respectively. ε_t is the error term which follows a $ARMA\,(1,1) - GARCH\,(1,1)$ process such that

$$\varepsilon_t = c + \phi_1 \varepsilon_{t-1} + \theta_1 \eta_{t-1} + \sigma_t \eta_t, \qquad \forall t \in \mathbb{Z}. \tag{83}$$

where c could be any constant, ϕ_1 is the $AR(1)$ coefficient, θ_1 is the coefficient of the corresponding innovations (i.e. $MA(1)$ process); $\eta_t \,\forall\, t \in \mathbb{Z}$ is a sequence of i.i.d. random variables such that $\eta_t \sim N\,(0,1)$ (i.e. as defined in Definition 2.1.3.).

There also exists nonnegative constants ,$\omega > 0$, $\alpha_1 \geq 0$ and $\gamma_1 \geq 0$ associated with the $GARCH(1,1)$ process such that

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \gamma_1 \sigma_{t-1}^2, \qquad \forall t \in \mathbb{Z}. \tag{84}$$

We generated the response variable, Y_t, with $t = 1, 2, \dots, T$ where $T = 200$. $\eta_t \,\forall\, t \in \mathbb{Z}$ are simulated from a standard normal distribution while the predictor variables, $X_t = (X_{t1}, X_{t2})'$, are similarly generated from a multivariate standard normal distribution. In order to test the hypothesis of the significance of β_2 as stated in (74), the true values of the regression coefficients were chosen to be

$$\beta_0 = (0.9,\, 0.01)' \tag{85}$$

and the variance parameter values were set as

$$\begin{cases} c &= 1, \\ \phi_1 &= 0.4, \\ \theta_1 &= 0.55, \\ \omega &= 0.1, \\ \alpha_1 &= 0.2, \\ \gamma_1 &= 0.7\,. \end{cases} \tag{86}$$

In order to compute the response variable, it was pertinent to initialise the ε_{t-1}, σ_t, σ_t^2 for $t = 0$; they were all set to 1. In Figure 8, we have a weekly time series plot of one simulation run of Y_t, given the information above and corresponding log returns of Y_t.

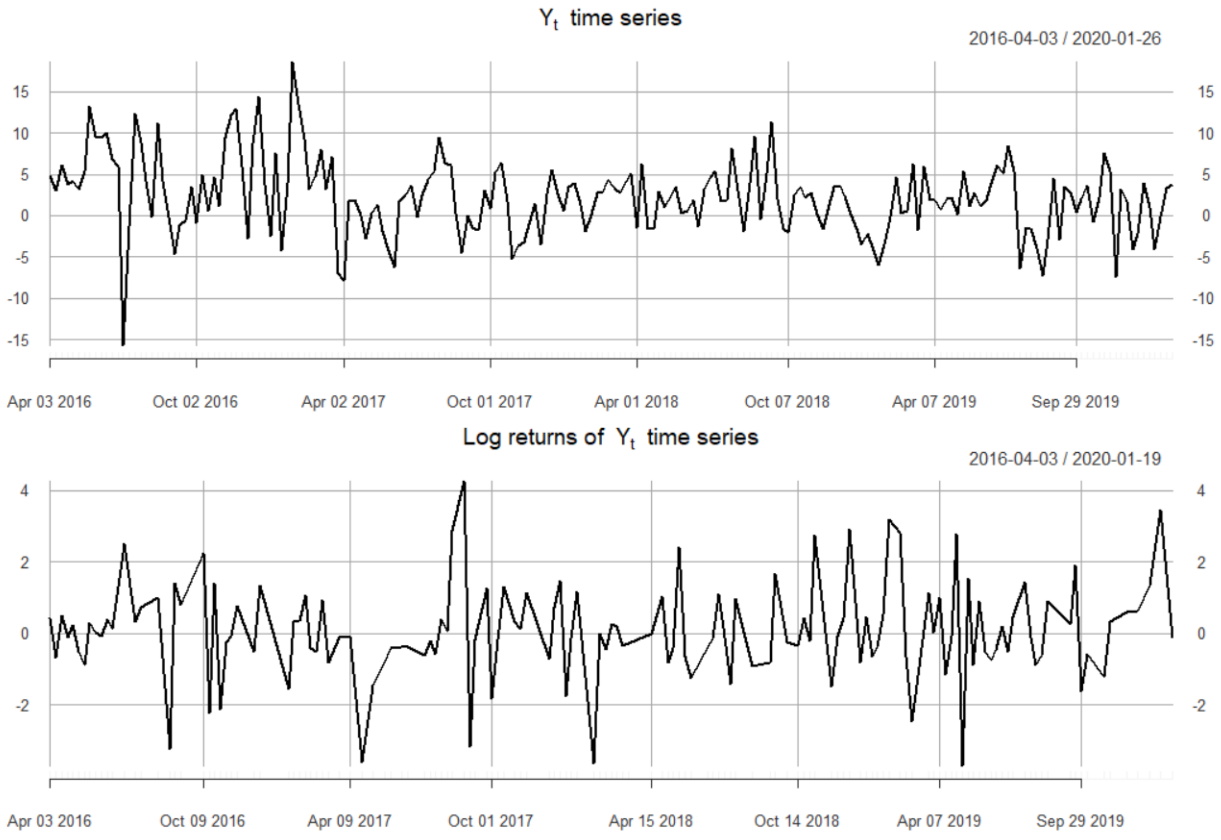


Figure 8: Weekly time plots of simulated Y_t and corresponding log returns of Y_t.

In Figure 8, Y_t and its corresponding log returns, $\log(Y_t)$ are displayed. There is evidence of volatility clustering depicted in the figure above and a lot of volatility can be seen particularly, from October 2016 to April 2017. There are large positive and negative shocks depicted in the price series that translate into the log returns series. It should be noted that some log returns of the price series could not be calculated as in Definition 2.1.1 (i.e. due to any negative values in the price series) and were hence omitted from the dataset before proceeding forward. This reduced the number of log return values but not by a significant amount. There are no visible upward or downtrends and the variability is centred around a mean of zero.

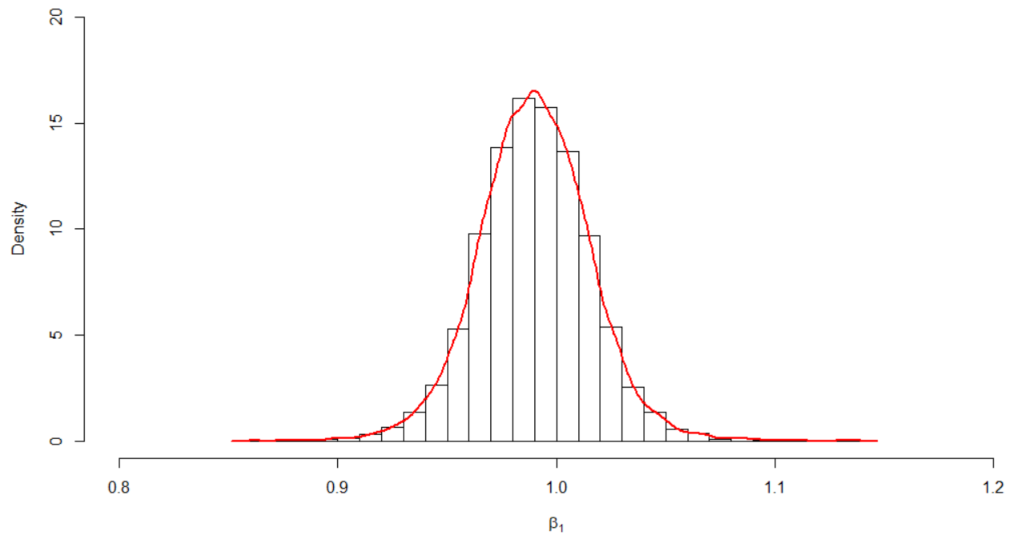
The simulated runs are each analysed and we then estimate the UR coefficients for the model in (82)-(84) by way of MLE. The MLE procedure uses the Newton-Raphson algorithm as a standard approach of estimation in the *maxLik* [27] package in R [14]. After estimating $\tilde{\beta}_{UR}$ from each simulation run (i.e. $N = 10000$), we computed the linear shrinkage parameter, $\hat{\beta}_S\left(\pi\right)$ by choosing a range of values for π, provided that the restricted estimator chosen was $\hat{\beta}_R = (1,\, 0)'$ using prior information. For the values of π, we first considered the range $\pi \in \{0, 0.01, 0.05\,(0.05)\,0.30\}$. This proved to be inadequate and we expanded the grid search such that $\pi \in \{0, 0.01, 0.05\,(0.05)\,0.95, 0.99, 1\}$. Furthermore, we tabulated the MSE values relative to each π value chosen. The best estimator is selected based on the π which provides the minimum MSE computed using (81). In the following table are each π value, the corresponding $(1 - \pi)$ value and the computed MSE.

Table 8: Linear shrinkage estimator, $\hat{\beta}_S\left(\pi\right)$ optimisation results for different values of π and corresponding MSE for $T = 200$ provided that the restricted estimator chosen was $\hat{\beta}_R = (1,\, 0)'$.

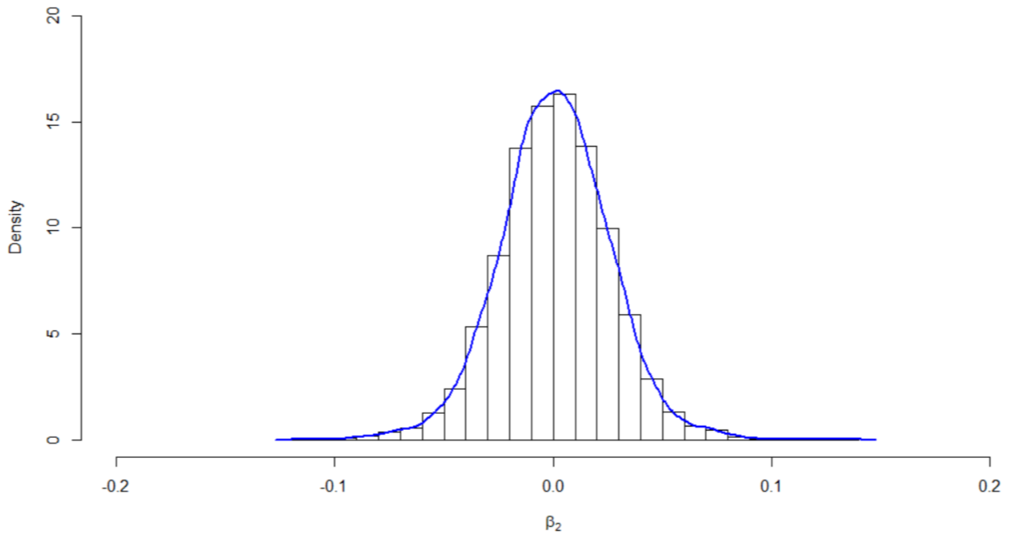
π	$1-\pi$	$MSE\left(\pi\right)$
0	1	0.0081
0.01	0.99	0.0079
0.05	0.95	0.0075
0.10	0.90	0.0072
0.15	0.85	0.0073
0.20	0.80	0.0078
0.25	0.75	0.0086
0.30	0.70	0.0098
0.35	0.65	0.0114
0.40	0.60	0.0133
0.45	0.55	0.0156
0.50	0.50	0.0183
0.55	0.45	0.0213
0.60	0.40	0.0247
0.65	0.35	0.0285
0.70	0.30	0.0327
0.75	0.25	0.0372
0.80	0.20	0.0420
0.85	0.15	0.0473
0.90	0.10	0.0529
0.95	0.05	0.0589
0.99	0.01	0.0639
1	0	0.0652

Table 8 displays the performance of each π in determining the best linear shrinkage estimator by way of the MSE provided that the restricted estimator chosen was $\hat{\beta}_R = (1,\, 0)'$. From the tabulated MSE values, the minimum MSE (i.e. $MSE = 0.007201651$) occurs when $\pi = 0.10$. The parameter π is referred to as the degree of confidence in the null hypothesis. Therefore as previously mentioned, if $\pi \to 0$ then there will be shrinkage and the restricted will subjugate the model, and if $\pi \to 1$ then there will be no degree of confidence in the restricted estimator, and the unrestricted model will be chosen over the restricted model. Given the results of our simulation study, we found that the optimal linear shrinkage estimator is when $\pi = 0.10$ which is closer to 0 than 1. This indicates that there is a greater dependency on the restricted model over the unrestricted model due to our prior knowledge and assumptions. There is some truth to this as the true value of β is $\beta_0 = (0.9,\, 0.01)'$ while the restricted estimator chosen was $\hat{\beta}_R = (1,\, 0)'$which are close in value.

In the following figure we have a histogram plots of the optimal shrinkage estimates given that it was found the a value of $\pi = 0.10$ gave the best results.



(i)



(ii)

Figure 9: Histogram of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(\pi)$ with fitted distribution curves: (i) $\hat{\beta}_{S,\,1}\,(0.1)$ and (ii) $\hat{\beta}_{S,\,2}\,(0.1)$

Figure 9 displays the distributions of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(\pi)$ from the simulation. Both linear shrinkage estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$ are shown to be asymptotically normally distributed and centred around means of their true values chosen which may suggest normality. The descriptive statistics are given in the table below to discern normality or non-normality.

Table 9: Basic statistics of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(0.1)$.

Basic Statistics	$\hat{\beta}_{S,\,1}\,(0.1)$	$\hat{\beta}_{S,\,2}\,(0.1)$
Minimum	0.861770	−0.117009
Maximum	1.136263	0.137265
1st Quartile (Q1)	0.973748	−0.014970
Median (Q2)	0.989780	0.000980
3rd Quartile (Q3)	1.006288	0.017232
Mean	0.989935	0.000993
Standard Deviation	0.025488	0.025520
Skewness	0.034650	0.046033
(Excess) Kurtosis	1.024060	1.042406

Given in the above table are the descriptive statistics of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(\pi)$. Both linear shrinkage estimates tend to have symmetric distributions with negligible positive skewness as they are quite close to a skewness value of 0. Their respective means and medians are also very close in value which suggests symmetry. Although, the excess kurtosis of both estimates given in Table 9, suggests they have heavier tailed distributions and hence non-normal.

We then looked at different restricted estimators to compute different set linear shrinkage estimators, $\hat{\beta}_{R\,(Scenario\,2)} = (0.001,\, 2)'$ and $\hat{\beta}_{R\,(Scenario\,3)} = (0.91,\, 0.02)'$ The other two scenarios considered are such that the new restricted estimators are chosen are such that none of the estimators are close to the true value (i.e. unlikely to be true) and the other being very close to the true estimators (i.e. very likely to be true).

Table 10 below displays the performance of each π in determining the best linear shrinkage estimator by way of the MSE for the restricted estimators chosen; $\hat{\beta}_{R\,(Scenario\,2)} = (0.001,\, 2)'$ and $\hat{\beta}_{R\,(Scenario\,3)} = (0.91,\, 0.02)'$. From the tabulated MSE values, the minimum MSE (i.e. $MSE = 0.06176855$) corresponds to a level of significance of $\pi = 0.95$ in the case of Scenario 2. This implies that the linear shrinkage estimator relies primarily on the estimates of the unrestricted model over those of the restricted model. This corresponds to our assumption as the restricted estimator, in this case, was chosen such neither of the estimators were close to the true value. In Scenario 3, we have that the smallest MSE (i.e. $MSE = 0.0003982787$) is obtained when the level of significance chosen is $\pi = 0.01$. This indicates that there little to no effect of the unrestricted model estimates as compared to the restricted model estimators. This due to the fact that the restricted estimators chosen, were very close to the true value. Hence the dependency of the restricted model over the unrestricted.

Table 10: Linear shrinkage estimator, $\hat{\beta}_S(\pi)$ optimisation results for different values of π and corresponding MSE for $T = 200$ for the restricted estimators chosen as $\hat{\beta}_{R\,(Scenario\,2)} = (0.001,\, 2)'$ and $\hat{\beta}_{R\,(Scenario\,3)} = (0.91,\, 0.02)'$.

π	$1-\pi$	$MSE_{\hat{\beta}_{R\,(Scenario\,2)}}(\pi)$	$MSE_{\hat{\beta}_{R\,(Scenario\,3)}}(\pi)$
0	1	1.1903	0.000400
0.01	0.99	1.1666	0.000398
0.05	0.95	1.0743	0.000523
0.10	0.90	0.9646	0.000974
0.15	0.85	0.8612	0.001753
0.20	0.80	0.7641	0.002860
0.25	0.75	0.6733	0.004296
0.30	0.70	0.5888	0.006060
0.35	0.65	0.5105	0.008153
0.40	0.60	0.4386	0.010573
0.45	0.55	0.3729	0.013322
0.50	0.50	0.3135	0.016400
0.55	0.45	0.2604	0.019805
0.60	0.40	0.2136	0.023539
0.65	0.35	0.1730	0.027601
0.70	0.30	0.1388	0.031991
0.75	0.25	0.1108	0.036710
0.80	0.20	0.0891	0.041757
0.85	0.15	0.0737	0.047132
0.90	0.10	0.0646	0.052835
0.95	0.05	0.0618	0.058867
0.99	0.01	0.0640	0.063929
1	0	0.0652	0.065227

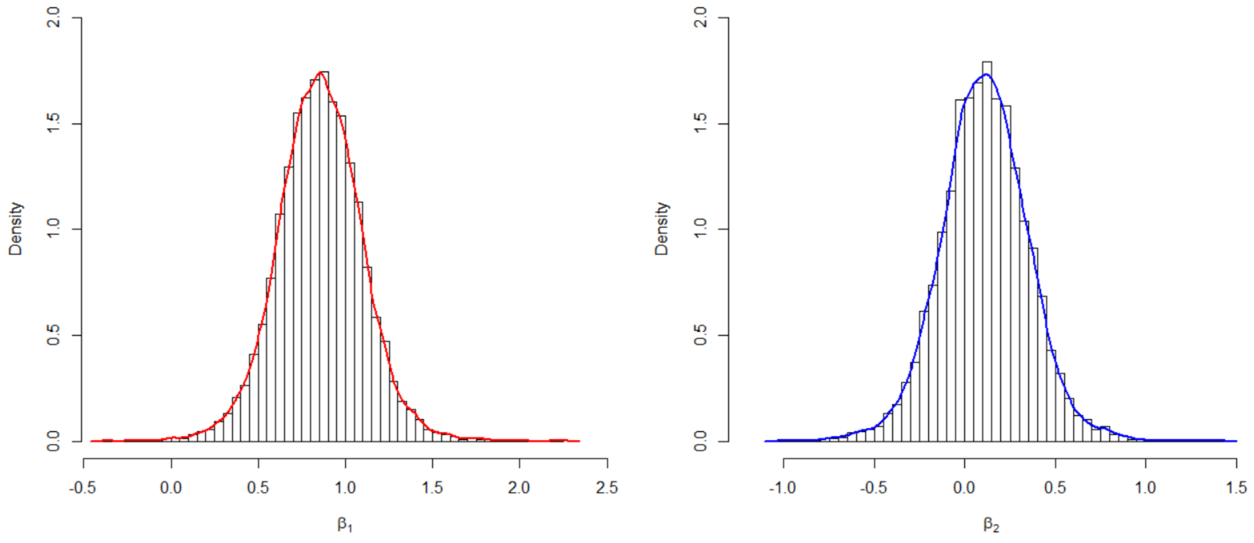


Figure 10: Histogram of the optimal linear shrinkage estimates, $\hat{\beta}_S(\pi)$ with fitted distribution curves: $\hat{\beta}_{S,\,1}(0.01)$ and $\hat{\beta}_{S,\,2}(0.01)$ under the restricted estimators chosen as $\hat{\beta}_{R\,(Scenario\,2)} = (0.001,\, 2)'$.

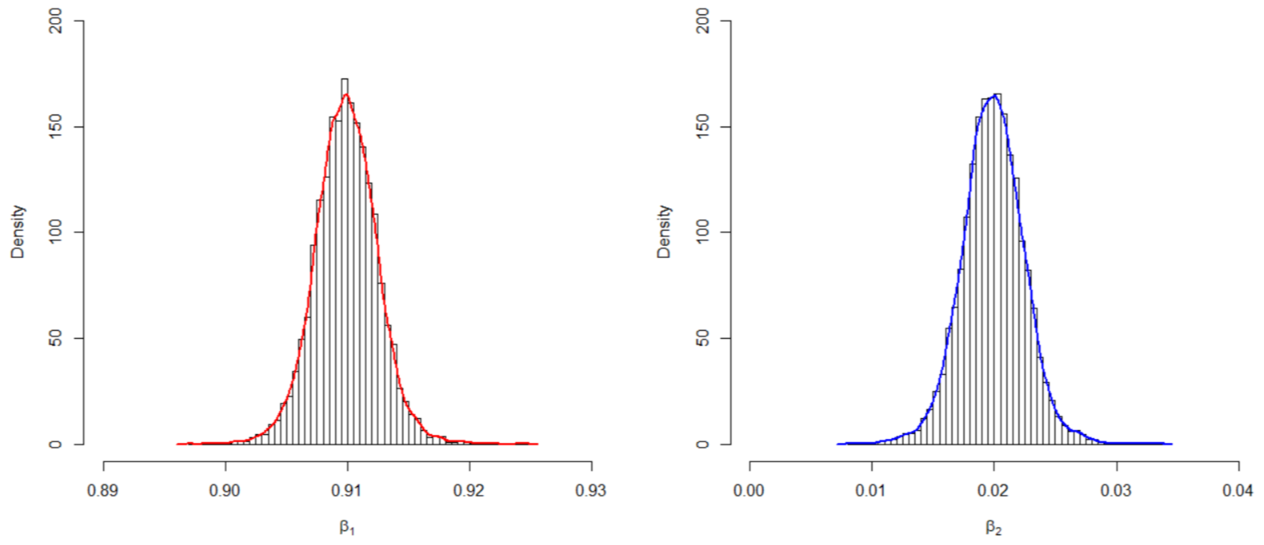


Figure 11: Histogram of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(\pi)$ with fitted distribution curves: $\hat{\beta}_{S,\,1}\,(0.95)$ and $\hat{\beta}_{S,\,2}\,(0.01)$ under the restricted estimators chosen as $\hat{\beta}_{R\,(Scenario\,3)} = (0.91,\,0.02)'$.

Figures 10 and 11 displays the distributions of the optimal linear shrinkage estimates, $\hat{\beta}_S\,(\pi)$ from the simulation of the two scenarios previously indicated. Both linear shrinkage estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$ are shown to be asymptotically normally distributed and centred around means of their true values chosen which may suggest normality. The descriptive statistics are given in the table below to discern normality or non-normality.

Table 11: Basic statistics of the optimal linear shrinkage estimates,$\hat{\beta}_{S,\,(Scenario\,2)}\,(0.01)$ and $\hat{\beta}_{S,\,(Scenario\,3)}\,(0.01)$.

Basic Statistics	$\hat{\beta}_{S,\,1;(Scenario\,2)}\,(0.95)$	$\hat{\beta}_{S,\,2;(Scenario\,2)}\,(0.95)$	$\hat{\beta}_{S,\,1;(Scenario\,3)}\,(0.01)$	$\hat{\beta}_{S,\,2;(Scenario\,3)}\,(0.01)$
Minimum	−0.363133	−1.011588	0.897077	0.008099
Maximum	2.244545	1.404022	0.924526	0.033527
1st Quartile (Q1)	0.700652	−0.042219	0.908275	0.018303
Median (Q2)	0.852961	0.109313	0.909878	0.019898
3rd Quartile (Q3)	1.009788	0.263705	0.911529	0.021523
Mean	0.854436	0.109433	0.909894	0.019899
Standard Deviation	0.242138	0.242439	0.002549	0.002552
Skewness	0.034650	0.046033	0.034650	0.046033
(Excess) Kurtosis	1.024060	1.042406	1.024060	1.042406

Given in the above table are the descriptive statistics of the optimal linear shrinkage estimates,$\hat{\beta}_{S;(Scenario\,2)}\,(\pi)$ and $\hat{\beta}_{S;(Scenario\,3)}\,(\pi)$. In both scenarios the linear shrinkage estimates tend to have symmetric distributions with negligible positive skewness as they are quite close to a skewness value of 0. Their respective means and medians are also very close in value which suggests symmetry. Although, the excess kurtosis of all estimates given in Table 11, suggests they have heavier tailed distributions and hence non-normal.

Alternatively, to linear shrinkage estimation, we consider preliminary test estimation (PTE), following Saleh [47]. In the PTE, we require a range of α values to test the hypotheses in (74). We considered the set $\alpha \in 0\{, 0.01, 0.05\,(0.05)\,0.30\}$, levels of significance to be tested. However, since the variance of the model plays a significant role in the computation of the test statistic, we would be able to better capture a trend with a larger spread of levels of significance. Hence we considered the following $\alpha \in \{0.1,\ 0.25,\ 0.50\}$. Since we have simulated and estimated $\tilde{\beta}_1$ and $\tilde{\beta}_2$, $N = 10000$ times, we can use those bootstrapped estimates to compute a $100\,(1-\alpha)\,\%$ bootstrap confidence intervals for each level of significance, for each unrestricted estimate, $\tilde{\beta}_{UR}(\alpha)$.

Table 12: The $100\,(1-\alpha)\,\%$ confidence intervals of the unrestricted estimates, $\tilde{\beta}_{UR}$ for $\alpha \in \{0.1,\ 0.25,\ 0.50\}$ levels of significance.

Level Of Significance	$100\,(1-\alpha)\,\%$ Confidence Interval of $\tilde{\beta}_{UR_1}(\alpha)$			$100\,(1-\alpha)\,\%$ Confidence Interval of $\tilde{\beta}_{UR_2}(\alpha)$	
	Lower Limit (LL)	Upper Limit (UL)		Lower Limit (LL)	Upper Limit (UL)
$\alpha = 0.10$	0.8951615	0.9035465		0.005731366	0.01412663
$\alpha = 0.25$	0.8964220	0.9022860		0.006993331	0.01286467
$\alpha = 0.50$	0.8976348	0.9010732		0.008207716	0.01165028

We can, therefore, determine whether the restricted or unrestricted model will be chosen for each case. We have decided to use the bootstrapping confidence interval method to test our hypothesis due to the complexity in the computation of the information matrix given in (79) which is further derived in Appendix A. So in turn, if zero is found within the confidence interval of $\tilde{\beta}_2$, then we do not reject the null hypothesis (i.e. restricted model estimates are chosen over the unrestricted model estimates) as shown in (75) for a particular level of significance, α. From the table above, there is evidence that as the level of significance tends to increase, the confidence interval lengths tend to narrow and move away from the hypothesised value, which increases the chance of rejecting the null hypothesis. Given the hypothesis in (74), the hypothesised value of zero cannot be found within the confidence interval limits of $\tilde{\beta}_2$, even at a 10% level of significance. Therefore we reject the null hypothesis that $\beta_2 = 0$, at all levels of significance and conclude that β_2 is significantly different from zero. This then means that the PTE is chosen to have the estimates of the unrestricted estimator, $\tilde{\beta}_{UR} = \left(\tilde{\beta}_1,\ \tilde{\beta}_2\right)'$. All the figures and tables produced have been computed in R [14] and the relevant code can be found in Appendix C.

5.3.2 Bitcoin price data analysis

The data used to apply the theory developed in the previous sections includes the Bitcoin returns used in Chapter 3, converted from daily data to weekly data; this will be considered the dependent variable of the regression model. It was previously determined in a recent paper by Sovbetov [51] that attractiveness/interest was a contributing factor of significance, influencing cryptocurrency prices.

Consequently, one of the corresponding predictor (i.e independent) variables considered was a dataset of Google searches with regards to the term "bitcoin" and associated terms. We obtained 104 weekly observations from the Google Trends [53] database between the period of March, 31st, 2016 to March, 31st, 2018; the same time period considered for the Bitcoin data. The data values recorded represent search interest of the term "bitcoin" relative to the highest peak on the chart worldwide within the specified time frame. The values provided are a rating scale scores ranging from 0 to 100 such that when the term search is at a peak in popularity, a value of 100 is recorded; a score of 50 means that the term has half the interest; and finally a rating of 0 indicates that there was probably an insufficient amount data for the particular term searched at the time. We considered this information as it could be believed that provided that we are the Internet age, that Google users' search behaviour of the term "bitcoin" and associated terms, may be in indicator of interest that may influence the price behaviour of Bitcoin. Note that all the figures and tables produced have been computed in R [14] and the relevant code can be found in Appendix D.

The choice of our second predictor variable was sparked by a determination realised in the paper released by Sovbetov [51]. He stated and hence determined in this paper that the traditional financial market has no bearing influence on the cryptocurrency market, hence would have little to no effect on the price of Bitcoin. Therefore, similarly to the study done by Sovbetov [51], we chose the S&P 500 index to test this theory. The S&P 500 index is deemed the most commonly followed stock market equity index that measures the stock performance of 500 of the largest companies listed on the stock markets in the United States. The S&P 500 index data used in our analysis, were collected between the period of March, 31st, 2016 to March, 31st, 2018 from the *quantmod* [46] package in R [14], which sources its data from Yahoo Finance. There is a total of 104 observations of weekly price trades for each of the 3 variables considered. Time series plots of our data are also displayed in the following figures.

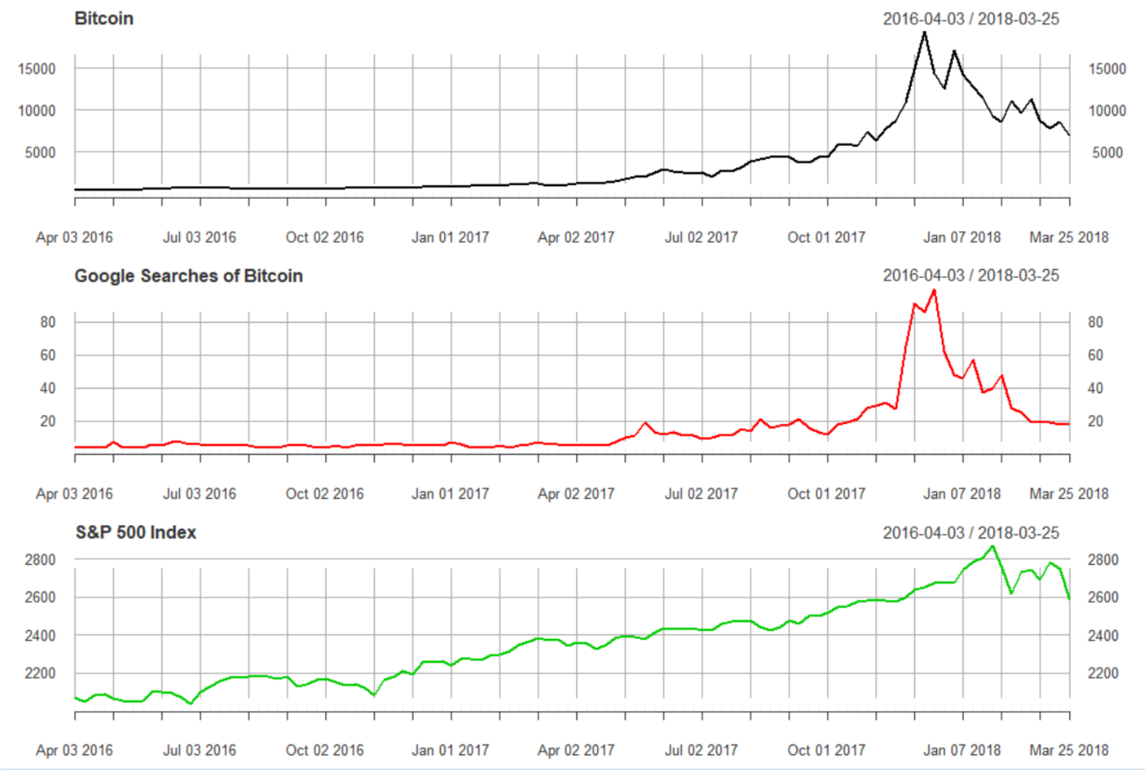


Figure 12: Time plots the USD/BTC weekly rate, weekly Google searches in and around the term "bitcoin" and S&P 500 index weekly rate.

Figure 12 is a time series plot of the variables under consideration for the modelling. The Bitcoin price series of weekly data show a constant variability from April 2016 to April 2017. We then notice an increasing trend until about November 2017 which is the period of time in which Bitcoin gained a lot of traction and cryptocurrencies became a topic of interest even on the news. During the months of December and January, large spikes in the price can be seen. This volatility remained up until the end of the year, even though the price began to drop quite rapidly. A similar trend to that of the USD/BTC weekly rate is also depicted in the Google searches of "Bitcoin" time plot. This suggests that these two variables under consideration, in particular, are likely to correlated in some way. The interest and then lack thereof in the cryptocurrency may have been a large contributing factor that led to its rapid rise and fall of the Bitcoin price over that period. The S&P 500 index weekly rate time plot also displayed in Figure 12 shows a gradually increasing trend from the sample period taken. Some volatility and positive spikes can also be seen towards the end of January 2018, through February until March 2018. Ultimately we would like to test and therefore determine whether there is a link between the cryptocurrency (i.e. Price of Bitcoin) and traditional financial markets (i.e. S&P 500 index).

As has been mentioned in the data description, the preliminary test will be hypothesised as follows:

$$\begin{aligned} H_0 &: \quad \beta_2 = 0 \\ H_A &: \quad \beta_2 \neq 0 \end{aligned} \tag{87}$$

where β_2 is the regression coefficient of the S&P 500 index and consequently, β_1 is the regression coefficient of the Google searches. This test will determine whether the traditional financial market represented by the S&P 500 index, has any significant impact on the price of Bitcoin.

As previously done in the simulation study, for choosing α (i.e. level of significance) in the preliminary test, we again consider the set $\alpha \in \{0.01, 0.05\,(0.05)\,0.50\}$ in order to possibly capture a trend. Bootstrap samples were taken and the unrestricted regression coefficients were estimated from each sample. Again, due to the complexity in the computation of the information matrix given in (79) which is further derived in Appendix A, used to formulate the test statistic; the bootstrapping confidence interval method was used to test our hypothesis. It was then determined that if zero is not found within the confidence interval of β_2, then we do not reject the null hypothesis (i.e. restricted model estimates are chosen over the unrestricted model estimates) as shown in (75) for a particular level of significance, α.

The linear shrinkage estimator was also considered in this application. In order to choose the optimal parameter π (i.e. degree of confidence in the null) that provides the best linear shrinkage estimator, $\hat{\beta}_S\,(\pi)$, we again use bootstrap sampling. The unrestricted estimates of the regression coefficients are computed from each bootstrap sample. The restricted estimator is computed as having the same estimated value for $\hat{\beta}_1$ as that of the "true" parameter β_{01} while the estimated value for the restricted $\beta_2 = 0$. It should be noted that the "true" estimator, β_0 used in this case was a simple estimation of the entire dataset. Provided that the unrestricted and restricted estimates have been computed, linear shrinkage estimator, $\hat{\beta}_S\,(\pi)$ can also be computed by way of (49) where π_i is the i-th value in a grid range of numbers chosen between 0 and 1. For each linear shrinkage estimate $\hat{\beta}_S\,(\pi_i)$, we compute the MSE as indicated in (81) over all replications. The optimal linear shrinkage estimator is then selected provided that it has the smallest MSE for a specified π_i in the grid.

In order to obtain the bootstrap samples for the estimation procedures, we took block bootstrap samples of the that consisted of $n_1 = 73$ (i.e. 70% of the observations) for the training set and $n_2 = 31$ (i.e. 30% of the observations) for the testing set; test set not used. The bootstrap training samples were taken 10000 times with replacement from data matrix $(y_t,\, x_{t1},\, x_{t2})$ and used to estimate the regression coefficients. We examined the point estimates, standard errors and mean square errors of the estimates and then calculate the estimators, SE, and the perform the preliminary test for each bootstrap replication. Finally, the estimator will be selected that has minimum MSE over all replications in the case of linear shrinkage, whilst the unrestricted or restricted estimator will be chosen based on whether the hypothesised value of β_2 is either found or not within the bootstrap confidence intervals.
Before an $ARMA-GARCH$ process could be modelled to the errors of the regression model in (45), we first needed to estimate the mean process, $X_t'\beta$ of the regression model by way of Ordinary Least Squares (OLS) estimation. This then allowed us to isolate the errors (i.e. residuals). A plot of the residuals is visualised in the following figure.

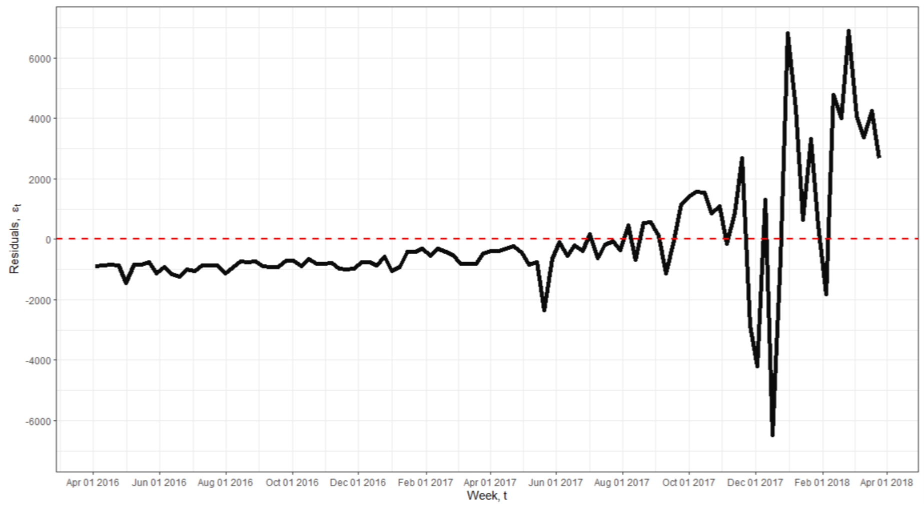


Figure 13: OLS residuals time plot.

We then tested for stationarity using the Augmented Dickey-Fuller (ADF) test on the OLS residuals and it was found that at 5% significance level, we reject the null hypothesis of a unit-root, therefore we conclude that the data is stationary. This is confirmed by the time plot displayed in Figure 13 which depicts that the time series is centered around a constant negative mean. The basic statistics reveal that the OLS residuals are skewed to the right and extremely leptokurtic, having a very high excess kurtosis. The basic statistics and ADF test results of the OLS residuals are reported in the following table.

Table 13: Basic statistics and ADF test results of OLS residuals.

Basic Statistics	OLS Residuals
Minimum	-6486.902
Maximum	6908.974
1st Quartile (Q1)	-865.8445
Median (Q2)	-655.9811
3rd Quartile (Q3)	-2.788821
Mean	-88.8285
Standard Deviation	1860.28
Skewness	1.261413
(Excess) Kurtosis	4.284915

ADF results	OLS Residuals
Test statistic	-3.9528
p-value	0.01417

We further analysed the residuals to determine if they follow a Gaussian or non-Gaussian distribution. This was done by following the same procedure as was performed in the previous chapter on the Bitcoin log returns, prior to the model fitting. Firstly, we performed a basic 2-sided t-test as well as tests under the assumption of normality such as the test for skewness, kurtosis and Jarque-Bera (JB) test.

Table 14: OLS residuals distribution normality tests.

	Two sided t-test	Skewness Test	Kurtosis Test	Jarque-Bera Normality Test
Test statistic	-0.48696	1.2798	7.4271	113.32
p-value	0.6273	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$

From the table above, we do not reject the null hypothesis that the true mean of the residuals is zero at a 5% level of significance. Meanwhile the sample estimates of $x = -88.8285$; this is significantly different from zero. Although, the 95% confidence interval is given by $(-450.6065, 272.9495)$. The skewness normality test also fails to reject the null hypothesis and as a result the kurtosis test had to be performed revealing that our residuals series is indeed leptokurtic; we reject the null hypothesis with 95% confidence that the distribution is not normal. This is reaffirmed by the JB-test that combines the tests for skewness and kurtosis as we reject the null hypothesis of normality even at a 1% level of significance. The skewness and kurtosis tests are extremely sensitive and the unusual results from the skewness test may be due to possible outliers, and considering we are working with a small concentrated data set.

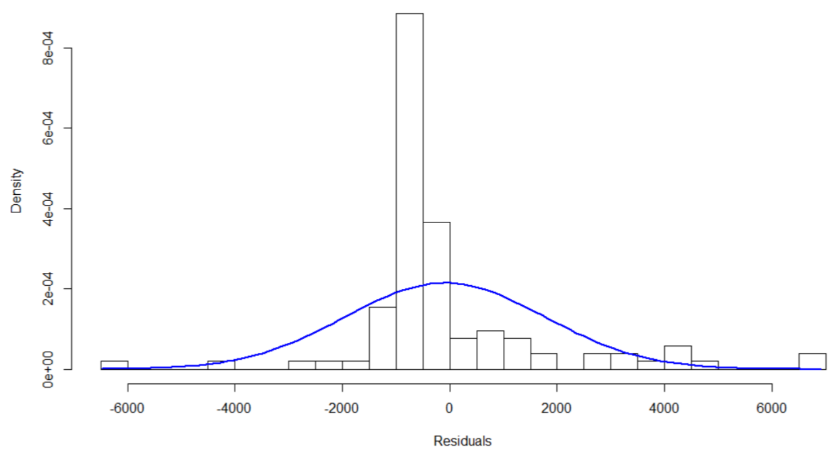


Figure 14: Histogram of the OLS residuals with a fitted normal distribution curve.

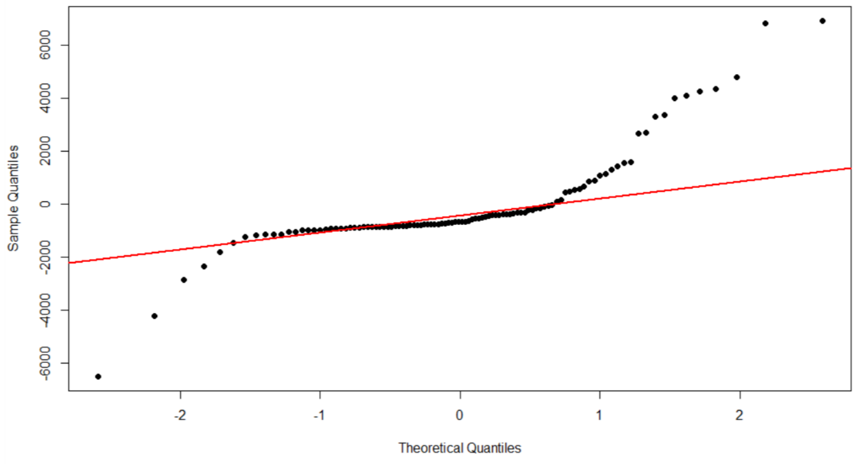


Figure 15: Q-Q plot of the OLS residuals distribution.

From Figures 14 and 15, the OLS residuals data exhibits attributes that lead us to conclude that it can be modelled by a non-Gaussian distribution. It can be seen from the histogram that the residuals are slightly skewed to the right and the data having a skewness of 1.2614 as reported in Table 13. The residuals are also extremely leptokurtic (i.e. very heavy tails) as is shown in the quantile-quantile (Q-Q) plot and the residuals have a very high excess kurtosis of 4.2849. From the analysis done above, we suggest modelling the residuals to an $ARMA-GARCH$ process with a non-normal underlying distribution.

Given the ambiguity of the tests of normality on the OLS residuals, it was noted that the OLS residuals were extremely spread with a standard deviation of $s = 1860.28$. Therefore prior to fitting of an $ARMA - GARCH$ process to the OLS residuals, we saw fit to remove any outliers from the data in order to obtain better parameter estimates in the final model. In order to remove the outliers, we firstly determined the outlier limits of the residuals using the following formulae:

$$
\begin{aligned}
LL_{outlier} &= Q_1 - (1.5 * (Q_3 - Q_1)) \\
UL_{outlier} &= Q_3 + (1.5 * (Q_3 - Q_1))
\end{aligned}
$$

where $LL_{outlier}$ and $UL_{outlier}$ are the lower and upper outlier limits, respectively, and Q_1 and Q_3 correspond to the lower and upper quartiles of the OLS residuals. As a result, any OLS residuals found to be are less than $LL_{outlier}$ or greater than $UL_{outlier}$ were deemed outliers. Consequently, the OLS outliers and their corresponding (y_t, x_{t1}, x_{t2}) were also removed from the data set and the remaining observations were then used in the final analysis. This resulted in a drop in the number of weekly observations from 104 to 85 which then also reduced the sample size of our bootstrap samples which now consisted of $n_1 = 60$ (i.e. 70% of the observations) for the training set and $n_2 = 25$ (i.e. 30% of the observations) for the testing set; again test set not used. The OLS residuals new time plot is given as follows:

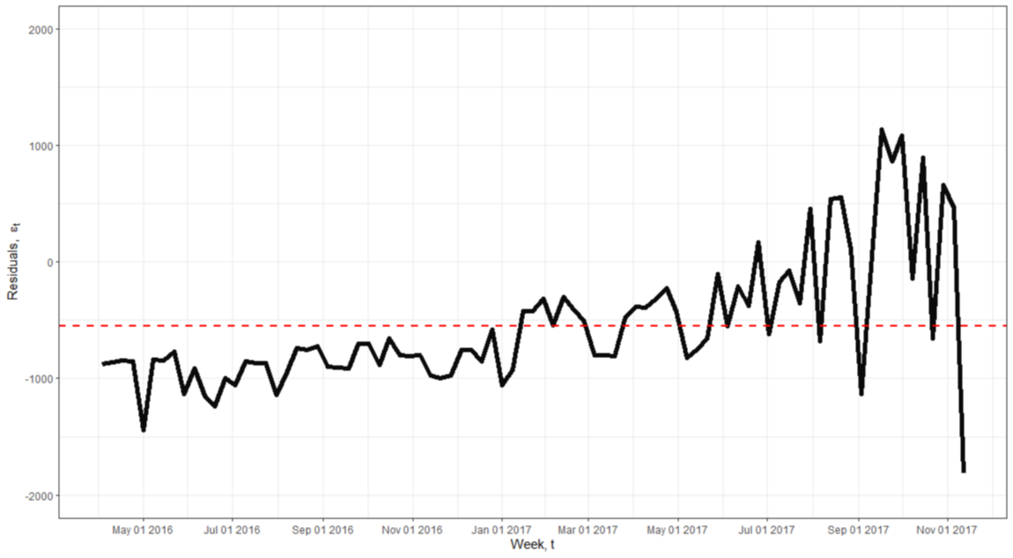


Figure 16: New OLS residuals time plot.

Table 15: Basic statistics of the new OLS residuals.

Basic Statistics	OLS Residuals
Minimum	-1811.266067
Maximum	1140.117849
1st Quartile (Q1)	-868.558955
Median (Q2)	-740.222232
3rd Quartile (Q3)	-355.720401
Mean	-546.044621
Standard Deviation	559.679999
Skewness	1.157457
(Excess) Kurtosis	1.284560

The table and figure above of the new OLS residuals show that all basic statistics have seen a reduction due to the change in the spread of the data, while keeping the original nature of the data. This has also greatly reduced the excess kurtosis, although the new OLS residuals still remained leptokurtic.

After the extraction and empirical analysis of the OLS model residuals, we then fit several different $ARMA\,(p,\,q)-GARCH\,(m,\,n)$ models to the residuals as in (46) and (47). Even though the results of the empirical analysis suggested that the underlying distribution was non-normal, keeping inline with the theory developed, the different $ARMA-GARCH$ models fit to the OLS residuals all had underlying standard normal distributions (i.e. i.i.d. $\eta_t \sim N\,(0,1)$). The best fitted model was chosen based on the information criterion results produced by way of the *rugarch* [22] in R [14] software.

Table 16: Model fitting results (AIC, BIC, SIC, $HQIC$) of different $ARMA\,(p,\,q)-GARCH\,(m,\,n)$ models.

	AIC	BIC	SIC	HQIC
$ARMA(0,0)-GARCH(1,1)$	14.796	14.911	14.792	14.542
$ARMA(1,0)-GARCH(1,1)$	14.485	14.628	14.478	16.046
$ARMA(0,1)-GARCH(1,1)$	14.618	14.761	14.611	14.675
$ARMA(1,1)-GARCH(1,1)$	14.393	14.566	14.384	14.463
$ARMA(2,1)-GARCH(1,1)$	14.372	14.573	14.360	14.453
$ARMA(1,2)-GARCH(1,1)$	14.389	14.590	14.377	14.470
$ARMA(2,2)-GARCH(1,1)$	14.254	14.484	14.238	14.346

From the model fitting results in Tables 16, it was found that the best fitted model to the residuals, with the smallest information criterion values was an $ARMA(2,2)-GARCH(1,1)$ under a standard normal distribution. The model of the residuals is given as follows:

Model. Given the model's parameter estimates, the $ARMA(2,2)-GARCH(1,1)$ model with $\eta_t \sim N\,(0,1)$ (i.e. the mean and variance are standardised to be $\mu=0$ and $\sigma^2=1$) as the conditional distribution, is defined as follows

$$\begin{aligned}\hat{\epsilon}_t &= -568.62126-0.02301\varepsilon_{t-1}+0.97133\varepsilon_{t-2}+\eta_t+0.56286\eta_{t-1}-0.67524\eta_{t-2}\\ \hat{\sigma}_t^2 &= 1342.14876+0.24635\varepsilon_{t-1}^2+0.75265\sigma_{t-1}^2\end{aligned}$$

After obtaining these results we substitute the above model of the residuals into our full model to re-estimate the regression coefficients. By way of bootstrap sampling, we obtained 10000 estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$. It should be noted that prior to the removal of the outliers in the data set, we experienced some difficulty in the maximum likelihood estimation (MLE) of the estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$; "NaN" values were produced for the standard error, $t-$statistic and $p-$values with respect to either of the regression coefficient estimates in some cases. This was due to the difficulties experienced in the computation of variance-covariance matrix. Consequently, this led to a very large spread of both regression coefficient estimates which we deemed unreliable. Therefore to obtain more reliable results, the outliers were eliminated from the data set which allowed us to proceed.

The final distributions of the estimates computed from the revised data set are displayed in the following figures.

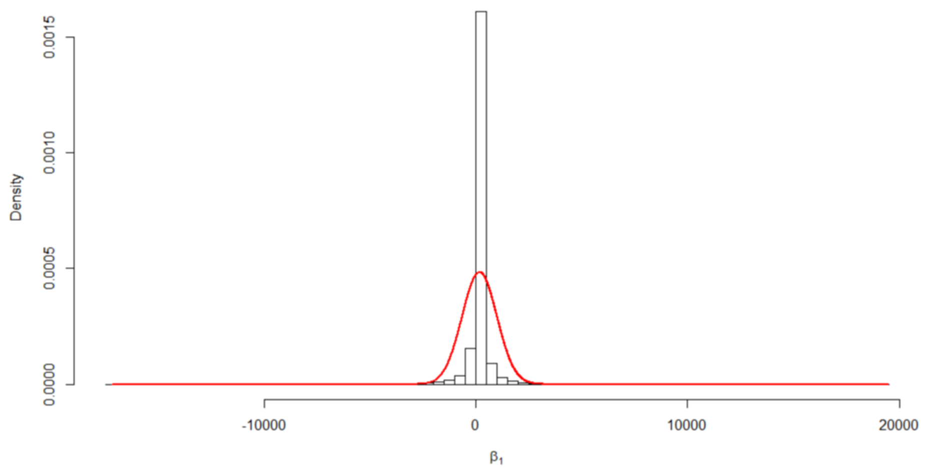


Figure 17: Histogram of the unrestricted $\tilde{\beta}_1$ estimates with a fitted normal distribution curve.

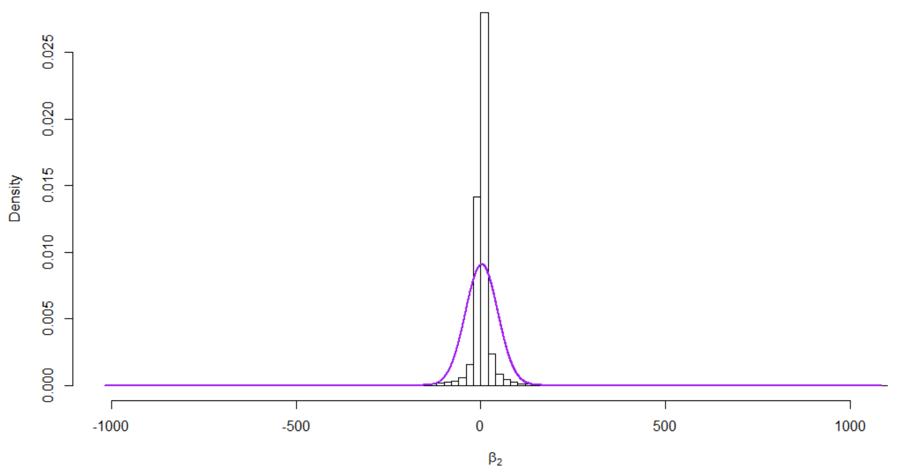


Figure 18: Histogram of the unrestricted $\tilde{\beta}_2$ estimates with a fitted normal distribution curve.

From Figures 17 and 18, the regression coefficient estimates have heavy tailed distributions (i.e. long tails) and centred around their respective means; this suggests symmetry. The basic statistics of the estimates given below, lead us to conclude that the distributions of the estimates produced, are also extremely leptokurtic (i.e. very heavy tails). This is also shown by ill fit of the normal distribution curves fitted in both figures. A table of the basic statistics is given below.

Table 17: Basic statistics of unrestricted regression coefficient estimates, $\tilde{\beta}_1$ and $\tilde{\beta}_2$.

Basic Statistics	$\tilde{\beta}_1$	$\tilde{\beta}_2$
Minimum	-17218.11	-1018.296
Maximum	19465.73	1083.477
1st Quartile (Q1)	121.6041	-1.587014
Median (Q2)	195.5588	1.170979
3rd Quartile (Q3)	251.0921	5.611158
Mean	179.8388	2.955703
Standard Deviation	823.0988	43.90382
Skewness	0.568777	0.862125
(Excess) Kurtosis	158.3622	133.1836

Table 17 displays the summary statistics of $\tilde{\beta}_1$ and $\tilde{\beta}_2$ regression coefficients estimated when fitting a linear regression with $ARMA - GARCH$ errors. As can be seen in the table above, the standard deviations for either estimate, are large due to the large variability in the estimation of the regression coefficients. The ranges of the unrestricted regression coefficient estimates for $\tilde{\beta}_1$ and $\tilde{\beta}_2$ are 36683.85 and 2101.773, respectively. These are large values, however, they do correspond to the large standard deviations seen in the above table. This affirms the variability of the estimates. This does cause some concern as this may suggest that the data was not modelled well.

If we then consider the middle 50% of each of the estimates, it shows a lot less variability as their respective interquartile ranges (IQR) are 129.4881 for $\tilde{\beta}_1$ and 7.198172 for $\tilde{\beta}_2$. This also indicates that the majority of the estimates are concentrated within the middle of the distribution; this can also be seen in the Figures 17 and 18.

The estimates produced also depict that both the distribution of $\tilde{\beta}_1$ and $\tilde{\beta}_2$ are slightly skewed to the right. This is also confirmed by the fact that the mean of the unrestricted regression coefficient estimates of $\tilde{\beta}_2$ ($i.e.\, \bar{x} = 2.955703$) being greater than the median ($i.e.\, q_{50} = 1.170979$) indicating a positive skew. While, oddly the mean of the unrestricted regression coefficient estimates of $\tilde{\beta}_1$ ($i.e.\, \bar{x} = 179.8388$) is less than the median ($i.e.\, q_{50} = 195.5588$) which indicates a negatively skewed distribution. It is known that there is a general relationship between the mean and median statistics that is somewhat related to the skewness of a unimodal distribution; a distribution is positively skewed when the mean is greater than the median, a distribution is negatively skewed when the mean is less than the median, and the when then mean and median are equivalent, we get a symmetric distribution.

Considering that the underlying distribution was assumed to be normal when computing the estimates, we probably would have attained better estimates if we had considered different non-Gaussian distributions that would've taken into account the leptokurtic behaviour and skewness in the data. Although we were unable to include that in this mini-dissertation, this may be considered in future research.

The following table gives the $100\,(1-\alpha)\,\%$ confidence intervals of each unrestricted regression coefficient estimates at different level of significance.

Table 18: The $100\,(1-\alpha)\,\%$ confidence intervals of the unrestricted estimates, $\tilde{\beta}_{UR}\,(\alpha)$ for $\alpha \in \{0.01,\, 0.05,\, (0.05),\, 0.50\}$ levels of significance.

	$100\,(1-\alpha)\,\%$ Confidence Interval of $\tilde{\beta}_{UR_1}\,(\alpha)$			$100\,(1-\alpha)\,\%$ Confidence Interval of $\tilde{\beta}_{UR_2}\,(\alpha)$	
Level Of Significance	Lower Limit (LL)	Upper Limit (UL)		Lower Limit (LL)	Upper Limit (UL)
$\alpha = 0.01$	158.6372	201.0404		1.824816	4.086590
$\alpha = 0.05$	163.7064	195.9712		2.095204	3.816202
$\alpha = 0.10$	166.3000	193.3776		2.233549	3.677857
$\alpha = 0.15$	167.9900	191.6876		2.323694	3.587712
$\alpha = 0.20$	169.2904	190.3872		2.393053	3.518353
$\alpha = 0.25$	170.3703	189.3073		2.450656	3.460750
$\alpha = 0.30$	171.3079	188.3697		2.500669	3.410737
$\alpha = 0.35$	172.1462	187.5314		2.545383	3.366023
$\alpha = 0.40$	172.9114	186.7662		2.586199	3.325207
$\alpha = 045$	173.6210	186.0566		2.624047	3.287359
$\alpha = 0.50$	174.2871	185.3905		2.659576	3.251830

The results in Table 18 indicate that as the level of significance increases, the $100\,(1-\alpha)\,\%$ confidence intervals tend to shrink and narrow, in either case of the estimates. Although the confidence intervals computed for $\tilde{\beta}_{UR_1}\,(\alpha)$ also tend to shift to the left, while the confidence intervals computed for $\tilde{\beta}_{UR_2}\,(\alpha)$ tend to shift to the right as α becomes larger.

The aim of this real data analysis was to determine the significance of the S&P 500 index on the price of Bitcoin. Since our hypothesised value for β_2 was zero (i.e. indicating the insignificance of S&P 500 index on the price of Bitcoin), it was found that the hypothesised value could not be found within any of confidence interval limits of $\tilde{\beta}_{UR_2}\,(\alpha)$, as displayed in 18. We, therefore, reject the null hypothesis that $\beta_2 = 0$, at all levels of significance and conclude that β_2 is significantly different from zero. This means that the PTE chosen, has the estimates of the unrestricted estimator, $\tilde{\beta}_{UR} = \left(\tilde{\beta}_1,\, \tilde{\beta}_2\right)'$. It is also concluded that traditional financial markets do have some impact on the price of Bitcoin. This contradicts the analysis by Sovbetov [51], who previously determined that the traditional financial market has no influence on the cryptocurrency market.

5.3.2.1 An example to choose π

A demonstration of linear shrinkage estimation is provided here. The computation of the linear shrinkage estimates requires the restriction we want to place and test, while the MSE computation also requires the "true" values of $\beta_1$and β_2. In order to make this possible, it was necessary to obtain all the components. The key issue faced was, just the mere adjustment of the seed value for the initialisation of some values, caused the "true" parameters estimated, to change. This meant that "true" parameters could be chosen randomly, and any results could be found. Since the "true" values of β_1 and β_2 are unknown, we obtained estimates of these values by computing the regression coefficients from the entire data set without resampling or shuffling of the data. We obtained unrestricted estimates which were representative of the "true" values of $\beta_1$and β_2; $\beta_{01} = 112.613$ and $\beta_{02} = 4.980402$. Therefore, the "true" parameter, β_0, used in this example are as follows

$$\beta_0 = \begin{pmatrix} 112.613 \\ 4.980402 \end{pmatrix} \tag{88}$$

The restricted estimator chosen was, $\hat{\beta}_R = \left(\tilde{\beta}_1, 0\right)'$, given that our prior information assumes that the impact of the S&P 500 index (i.e. traditional financial market) has no effect on the price of Bitcoin. After having placed the restriction, estimated the unrestricted coefficients and determined the "true" regression coefficients to be used, we are then able to compute the linear shrinkage estimator, $\hat{\beta}_S(\pi)$.

It was a challenge in determining the linear shrinkage estimates as the "true" parameters are required to determine the optimal estimates by way of the minimum MSE.

Table 19: Linear shrinkage estimator, $\hat{\beta}_S(\pi)$ optimisation results for different values of π and corresponding MSE for $T = 200$ provided that the restricted estimator chosen was $\hat{\beta}_R = \left(\tilde{\beta}_1, 0\right)'$.

π	$1-\pi$	$MSE(\pi)$
0	1	681298.4
0.01	0.99	680638.9
0.05	0.95	678004.9
0.10	0.90	674721.0
0.15	0.85	671446.8
0.20	0.80	668182.3
0.25	0.75	664927.5
0.30	0.70	661682.4
0.35	0.65	658447.0
0.40	0.60	655221.2
0.45	0.55	652005.1
0.50	0.50	648798.7
0.55	0.45	645602.0
0.60	0.40	642414.9
0.65	0.35	639237.6
0.70	0.30	636069.9
0.75	0.25	632911.9
0.80	0.20	629763.6
0.85	0.15	626624.9
0.90	0.10	623496.0
0.95	0.05	620376.7
0.99	0.01	617888.2
1	0	617267.1

As previously done in the simulation study, Table 19 displays the performance of each π in determining the best linear shrinkage estimator by way of the MSE for the restricted estimators chosen; $\hat{\beta}_R = \left(\tilde{\beta}_1, 0\right)'$. We note that the MSE values tabulated are very large and that the minimum MSE found was 617267.1. The minimum MSE found corresponds to a level of significance of $\pi = 1$. Since we are working on real data, the true parameter values are unknown. We, therefore, are unable to discern a conclusion about the significance of the S&P 500 index on the price of Bitcoin using this method of estimation. Hence, a simple example was used to demonstrate the linear shrinkage estimation technique.

6 Conclusion

In this mini-dissertation, we explored and discussed the properties of the hybrid $ARMA-GARCH$ model. An introduction of the cryptocurrency market with a focus on Bitcoin is given. This was followed by an in-depth analysis of a simple application of $ARMA-GARCH$ fitting on a sample of Bitcoin data and documented the results. It was found that an $ARMA-GARCH$ model, under a GED distribution, fitted the Bitcoin data well and hence proved to be better than an OLS model. Then we discussed linear regression models with GARCH errors and developed the theory of linear regression models with $ARMA-GARCH$ errors. Preliminary test estimators and further discussed the theory of shrinkage estimation concerning regression modelling were looked at. A simulated time series demonstrated how to conduct linear shrinkage estimation and preliminary test estimation on linear regression models with $ARMA-GARCH$ errors. We placed and tested the effect of different restrictions on the simulated data and computed different sets of linear shrinkage estimates. It was concluded that if the prior information is known with some certainty, then the linear shrinkage estimates tended to have a greater dependency on the restricted model over its unrestricted counterpart. Alternatively, if the prior information was uncertain, then the restriction placed would have less of an impact on the linear shrinkage estimates. We also found that both linear shrinkage estimates of $\hat{\beta}_1$ and $\hat{\beta}_2$ tended to be asymptotically normally distributed. This indicated that modelling the regression model with $ARMA-GARCH$ errors performed well. Due to the complexity in the computation of the PT test statistic, we chose to use bootstrap confidence intervals to test the hypothesis. There was evidence to suggest that as the level of significance tends to increase, the confidence limits tended narrow and move away from the hypothesised value, which increased the chance of rejecting the null hypothesis. The hypothesis was then rejected at all levels of significance and in conclusion that the PTE is chosen would take on the estimates of the unrestricted estimator. Further research is proposed to develop more explicit forms of the derivatives needed to compute the information matrix and the PT test statistic. The unrestricted estimates obtained from bootstrapping still gave very large values which increased the spread of the distribution and this could also be addressed in future studies.

This mini-dissertation was concluded with an application of preliminary test and linear shrinkage estimation on the Bitcoin data. The aim of this was to determine the effect of the traditional financial market on the cryptocurrency market by way of the S&P 500 index. We experienced challenges in determining the linear shrinkage estimates as the “true” parameters are required to determine the optimal estimates by way of the minimum MSE. The key issue we faced was, just the mere adjustment of the seed value for the initialisation of some values, caused the “true” parameters estimated, to change. This meant that “true” parameters could be chosen randomly, and any results could be found. As a result, we were also unable to discern a conclusion about the significance of the S&P 500 index on the price of Bitcoin using this method of estimation. Hence, a simple example was used to demonstrate the linear shrinkage estimation technique. This may be considered in future research. As previously mentioned, we used the bootstrap confidence interval approach to compute the PTE and test our hypothesis. It proved to be a more reliable way to test our hypothesis than the linear shrinkage method. This was because the “true” parameters were not required even though the test statistic could not be computed. The results indicated that with some level of confidence whether our hypothesis will be rejected or not rejected. Since our hypothesised value of zero cannot be found within the confidence interval limits of $\tilde{\beta}_2$, even at a 10% level of significance, we, therefore, reject the null hypothesis that $\beta_2 = 0$, at all levels of significance and conclude that β_2 is significantly different from zero. This then means that the PTE is chosen to have the estimates of the unrestricted estimator, $\tilde{\beta}_{UR} = \left(\tilde{\beta}_1, \tilde{\beta}_2\right)'$.

Other error distributions are suggested for future research in the linear regression modelling space and shrinkage estimation applications such as nonparametric and semiparametric $ARMA - GARCH$ error distributions and possibly an $AFRIMA - GARCH$ error distribution. This could better explain phenomena such as Bitcoin. In conclusion, as can be seen in this mini-dissertation that Bitcoin exchange rates exhibit numerous and fascinating statistical characteristics due to its extreme volatility and unpredictability. With further study and research, deeper analytics may be required to explain its phenomena.

7 Appendix

Appendix A.

All the derivatives that will be defined are required to compute the variance-covariance matrix which is the inverse of the information matrix. The first order derivatives (i.e. gradient) and the second order derivatives (i.e. Hessian) of the log-likelihood given in (57) and (58), will be computed with respect to the vector of unknown parameters, $\xi = (\beta', \, \delta')$.

1) First order derivatives The first order derivatives of the log-likelihood with respect to the vector of unknown parameters, ξ is also referred to as the gradient or the score. The gradient is computed as follows

$$G(\xi) = \frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \xi} \tag{89}$$

Since $\xi = (\beta', \, \delta')$ is of the dimensions $((k+2+p+q+m+n) \times 1)$, the gradient can then be further expressed as

$$G(\xi) = \begin{bmatrix} \frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \beta_1} \\ \frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \beta_2} \\ \vdots \\ \vdots \\ \frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \gamma_{m-1}} \\ \frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \gamma_m} \end{bmatrix} \tag{90}$$

Differentiating the log-density with respect to the variance parameters. Recall that the vector of unknown variance parameters are $\delta' = (\delta_1', \, \delta_2')$ such that $\delta_1' = (c, \, \phi_1, \, \phi_2, \ldots, \, \phi_p \, \theta_1, \, \theta_2, \ldots, \, \theta_q) \; : \; (1+p+q) \times 1$ and $\delta_2' =$

$(\omega,\, \alpha_1,\, \alpha_2, \ldots,\, \alpha_n\, \gamma_1,\, \gamma_2, \ldots,\, \gamma_m)\ :\ (1+m+n)\times 1$, is given by

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{T}}(\xi)}{\partial \delta} &= \frac{\partial\left(-\frac{1}{2}log\left(2\pi\right)-\frac{1}{2}log\left(h_t^2\right)-\frac{1}{2}\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{1}{2}\frac{\mu^2}{h_t^2}\right)}{\partial\delta} \\
&= \left[-\frac{1}{2}\frac{1}{2\pi}\frac{\partial\left(2\pi\right)}{\partial\delta}\right]-\left[\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right]-\left[\frac{1}{2}\left(Y_t-X_t'\beta\right)^2\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right] \\
&\quad -\left[\frac{1}{2}\frac{1}{h_t^2}\left(2\right)\left(Y_t-X_t'\beta\right)\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta}\right]+\left[\left(Y_t-X_t'\beta\right)\mu\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right] \\
&\quad +\left[\frac{\mu}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta}\right]+\left[\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta}\right]-\left[\frac{1}{2}\left(2\right)\mu\frac{1}{h_t^2}\frac{\partial\mu}{\partial\delta}\right]-\left[\frac{1}{2}\mu^2\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right] \\
&= -\left[\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right]+\left[\frac{1}{2}\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right]-\left[\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right]+\left[\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta}\right] \\
&\quad -\left[\frac{\mu}{h_t^2}\frac{\partial\mu}{\partial\delta}\right]+\left[\frac{1}{2}\frac{\mu^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right] \\
&= \frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\left\{-\frac{1}{2}+\left[\frac{1}{2}\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\right]-\left[\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}\right]+\frac{1}{2}\frac{\mu^2}{h_t^2}\right\}+\frac{\partial\mu}{\partial\delta}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\} \\
&= -\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\left\{1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}+\frac{\partial\mu}{\partial\delta}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}. \qquad (91)
\end{aligned}$$

where

$$\frac{\partial\left(h_t^2\right)}{\partial\delta} = \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^{p}\phi_i+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^{q}\theta_i^2+E\left[\sigma_t^2\right]-\mu^2\end{array}\right)}{\partial\left(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q,\,\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m\right)}. \qquad (92)$$

and $y_t = \eta_t = 0$ for $t < 0$, $x_t = 0$ for $t < 0$, $\phi_t = 0$ for $t > p$, and $\theta_t = 0$ for $t > q$.

Note that for the $ARMA-GARCH$ specification, the computation of $\frac{\partial h_t^2}{\partial\delta}$requires the computation of $\frac{\partial h_0^2}{\partial\delta}$. In order to calculated h_0^2, we do so by the averaging the pre-sample square residuals, ε_t^2, that do not depend on δ. Therefore when $t < 1$

$$\begin{aligned}
\sigma_t^2 &= \varepsilon_t^2 \\
&= \frac{1}{T}\sum_{r=1}^{T}\hat{\varepsilon}_r^2
\end{aligned}$$

where $\hat{\varepsilon}_r^2$ are the consistently estimated residuals. Therefore we then get the following

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\delta} &= \frac{\partial\sigma_t^2}{\partial\delta} \\
&= \frac{\partial\varepsilon_t^2}{\partial\delta} \\
&= \frac{\partial}{\partial\delta}\left[\frac{1}{T}\sum_{r=1}^{T}\varepsilon_r^2\right] \\
&= \frac{\partial}{\partial\delta}\left[\frac{1}{T}\sum_{r=1}^{T}\left(Y_r - X_r^{'}\beta\right)^2\right] \\
&= 0.
\end{aligned}$$

and when $t \geq 1$

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\delta} &= \frac{\partial\left(\begin{array}{c} c^2 + 2c\mu\sum_{i=1}^{p}\phi_i + E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] + 2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right] + \sum_{i=1}^{q}\theta_i^2 + E\left[\sigma_t^2\right] - \mu^2 \end{array}\right)}{\partial\left(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q,\,\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m\right)} \\
&= \left[\begin{array}{c} \frac{\partial\left(h_t^2\right)}{\partial c} \\ \frac{\partial\left(h_t^2\right)}{\partial\phi_1} \\ \vdots \\ \vdots \\ \frac{\partial\left(h_t^2\right)}{\partial\gamma_{m-1}} \\ \frac{\partial\left(h_t^2\right)}{\partial\gamma_m} \end{array}\right]
\end{aligned} \tag{93}$$

where the i-th elements of each group of parameters in (93) is given by

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial c} &= \frac{\partial\left(\begin{array}{c} c^2 + 2c\mu\sum_{i=1}^{p}\phi_i + E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] + 2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right] + \sum_{i=1}^{q}\theta_i^2 + E\left[\sigma_t^2\right] - \mu^2 \end{array}\right)}{\partial c} \\
&= 2c + 2\mu\sum_{i=1}^{p}\phi_i - 2\mu\frac{\partial\mu}{\partial c}.
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\phi_i} &= \frac{\partial\left(\begin{array}{c} c^2 + 2c\mu\sum_{i=1}^{p}\phi_i + E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] + 2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right] + \sum_{i=1}^{q}\theta_i^2 + E\left[\sigma_t^2\right] - \mu^2 \end{array}\right)}{\partial\phi_i} \\
&= 2c\mu + 2E\left[\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]\frac{\partial\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}}{\partial\phi_i} + 2\frac{\partial E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]}{\partial\phi_i} + 2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\phi_i} - 2\mu\frac{\partial\mu}{\partial\phi_i} \\
&= 2c\mu + 2E\left[\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]\varepsilon_{t-i} + 2\frac{\partial E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]}{\partial\phi_i} + 2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\phi_i} - 2\mu\frac{\partial\mu}{\partial\phi_i}.
\end{aligned}$$

$$\varepsilon_t = c + \phi_1\varepsilon_{t-1} + \phi_2\varepsilon_{t-2} + \ldots + \phi_p\varepsilon_{t-p} + \theta_1\eta_{t-1} + \theta_2\eta_{t-2} + \ldots + \theta_q\eta_{t-q} + \sigma_t\eta_t.$$

$$
\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\theta_i} &= \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^p\phi_i+E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^q\theta_i^2+E\left[\sigma_t^2\right]-\mu^2 \end{array}\right)}{\partial\theta_i} \\
&= \frac{\partial\left\{\sum_{i=1}^q\theta_i^2\right\}}{\partial\theta_i}+2\frac{\partial E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right]}{\partial\theta_i}+2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\theta_i} \\
&= 2\theta_i+2\frac{\partial E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right]}{\partial\theta_i}+2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\theta_i}.
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\omega} &= \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^p\phi_i+E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^q\theta_i^2+E\left[\sigma_t^2\right]-\mu^2 \end{array}\right)}{\partial\omega} \\
&= \frac{\partial E\left[\sigma_t^2\right]}{\partial\omega}+2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\omega} \\
&= \frac{\partial\left(\omega+\sum_{i=1}^n\alpha_iE\left[\varepsilon_{t-i}^2\right]+\sum_{j=1}^m\gamma_jE\left[\sigma_{t-j}^2\right]\right)}{\partial\omega}+2\frac{\partial E\left[\eta_t\left\{\omega+\sum_{i=1}^n\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^m\gamma_j\sigma_{t-j}^2\right\}\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\omega} \\
&= 1+2\frac{\partial E\left[\eta_t\left\{\omega+\sum_{i=1}^n\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^m\gamma_j\sigma_{t-j}^2\right\}\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\omega}.
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\alpha_i} &= \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^p\phi_i+E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^q\theta_i^2+E\left[\sigma_t^2\right]-\mu^2 \end{array}\right)}{\partial\alpha_i} \\
&= \frac{\partial\left(\omega+\sum_{i=1}^n\alpha_iE\left[\varepsilon_{t-i}^2\right]+\sum_{j=1}^m\gamma_jE\left[\sigma_{t-j}^2\right]\right)}{\partial\alpha_i}+2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\alpha_i} \\
&= E\left[\varepsilon_{t-i}^2\right]\frac{\partial E\left[\varepsilon_{t-i}^2\right]}{\partial\alpha_i}+\frac{\sum_{j=1}^m\gamma_j\partial E\left[\sigma_{t-j}^2\right]}{\partial\alpha_i}+2\frac{\partial E\left[\eta_t\left\{\omega+\sum_{i=1}^n\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^m\gamma_j\sigma_{t-j}^2\right\}\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\alpha_i}.
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\gamma_i} &= \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^p\phi_i+E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^q\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^q\theta_i^2+E\left[\sigma_t^2\right]-\mu^2 \end{array}\right)}{\partial\gamma_i} \\
&= \frac{\partial\left(\omega+\sum_{i=1}^n\alpha_iE\left[\varepsilon_{t-i}^2\right]+\sum_{j=1}^m\gamma_jE\left[\sigma_{t-j}^2\right]\right)}{\partial\gamma_i}+2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\gamma_i} \\
&= E\left[\sigma_{t-j}^2\right]\frac{\partial E\left[\sigma_{t-j}^2\right]}{\partial\gamma_i}+2\frac{\partial E\left[\eta_t\left\{\omega+\sum_{i=1}^n\alpha_i\varepsilon_{t-i}^2+\sum_{j=1}^m\gamma_j\sigma_{t-j}^2\right\}\left\{\sum_{i=1}^p\phi_i\varepsilon_{t-i}\right\}\right]}{\partial\gamma_i}.
\end{aligned}
$$

The following derivative is also necessary for the computation of (91). The first order derivative of the conditional mean, μ with respect to the variance parameters in δ, we then get the following

$$\begin{aligned}\frac{\partial\mu}{\partial\delta} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\left(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q,\,\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m\right)}\\ &= \begin{bmatrix}\frac{\partial\mu}{\partial c}\\ \frac{\partial\mu}{\partial\phi_1}\\ \vdots\\ \vdots\\ \frac{\partial\mu}{\partial\gamma_{m-1}}\\ \frac{\partial\mu}{\partial\gamma_m}\end{bmatrix}\end{aligned}\tag{94}$$

where the i-th elements of each group of parameters in (94) are given as follows

$$\begin{aligned}\frac{\partial\mu}{\partial c} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial c}\\ &= \frac{1}{1-\phi_1-\phi_2-\ldots-\phi_p}.\end{aligned}$$

$$\begin{aligned}\frac{\partial\mu}{\partial\phi_i} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\phi_i}\\ &= \left(1-\phi_1-\phi_2-\ldots-\phi_p\right)^{-1}\frac{\partial c}{\partial\phi_i}+c\,(-1)\left(1-\phi_1-\phi_2-\ldots-\phi_p\right)^{-2}\frac{\partial\left(1-\phi_1-\phi_2-\ldots-\phi_p\right)}{\partial\phi_i}\\ &= 0+c\,(-1)\left(1-\phi_1-\phi_2-\ldots-\phi_p\right)^{-2}(-1)\\ &= \frac{c}{\left(1-\phi_1-\phi_2-\ldots-\phi_p\right)^2}.\end{aligned}$$

$$\begin{aligned}\frac{\partial\mu}{\partial\theta_i} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\theta_i}\\ &= 0.\end{aligned}$$

$$\begin{aligned}\frac{\partial\mu}{\partial\omega} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\omega}\\ &= 0.\end{aligned}$$

$$\begin{aligned}\frac{\partial\mu}{\partial\alpha_i} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\alpha_i}\\ &= 0.\end{aligned}$$

$$\begin{aligned}\frac{\partial\mu}{\partial\gamma_i} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\gamma_i}\\ &= 0.\end{aligned}$$

Now we differentiate the log likelihood density function with respect to $\beta = (\beta_1, \beta_2, \ldots, \beta_k)^{\backprime}$ regression coefficients (i.e. mean parameters):

$$\begin{aligned}\frac{\partial\mathcal{L}_{\mathcal{T}}(\xi)}{\partial\beta} &= \frac{\partial\left(-\frac{1}{2}log\left(2\pi\right)-\frac{1}{2}log\left(h_t^2\right)-\frac{1}{2}\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}+\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}-\frac{1}{2}\frac{\mu^2}{h_t^2}\right)}{\partial\beta}\\ &= \left[-\frac{1}{2}\frac{1}{2\pi}\frac{\partial\left(2\pi\right)}{\partial\beta}\right]-\left[\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]-\left[\frac{1}{2}\left(Y_t-X_t^{'}\beta\right)^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\\ &\quad -\left[\frac{1}{2}\frac{1}{h_t^2}(2)\left(Y_t-X_t^{'}\beta\right)\frac{\partial\left(Y_t-X_t^{'}\beta\right)}{\partial\beta}\right]+\left[\left(Y_t-X_t^{'}\beta\right)\mu(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\\ &\quad +\left[\frac{\mu}{h_t^2}\frac{\partial\left(Y_t-X_t^{'}\beta\right)}{\partial\beta}\right]+\left[\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]-\left[\frac{1}{2}(2)\mu\frac{1}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]-\left[\frac{1}{2}\mu^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\\ &= -\left[\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{1}{2}\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]-\left[\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\\ &\quad +\left[\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]-\left[\frac{\mu}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]+\left[\frac{1}{2}\frac{\mu^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\\ &= -\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{1-\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}+\frac{\partial\mu}{\partial\beta}\left\{\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}. \qquad (95)\end{aligned}$$

where

$$\begin{aligned}\frac{\partial\left(h_t^2\right)}{\partial\beta} &= \frac{\partial\left(\begin{array}{c} c^2+2c\mu\sum_{i=1}^{p}\phi_i+E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right]+2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]\\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right]+\sum_{i=1}^{q}\theta_i^2+E\left[\sigma_t^2\right]-\mu^2\end{array}\right)}{\partial\left(\beta_1,\beta_2,\ldots,\beta_k\right)}\\ &= \begin{bmatrix}\frac{\partial\left(h_t^2\right)}{\partial\beta_1}\\ \frac{\partial\left(h_t^2\right)}{\partial\beta_2}\\ \vdots\\ \vdots\\ \frac{\partial\left(h_t^2\right)}{\partial\beta_{k-1}}\\ \frac{\partial\left(h_t^2\right)}{\partial\beta_k}\end{bmatrix} \qquad (96)\end{aligned}$$

provided that the error term of regression model can be rewritten as

$$\begin{aligned}\varepsilon_t &= Y_t - X_t^{'}\beta \\ &= Y_t - \beta_1 X_{t1} - \ldots - \beta_k X_{tk}.\end{aligned}$$

then it follows that the i-th element of the vector in (96) is given by

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\beta_i} &= \frac{\partial\begin{pmatrix} c^2 + 2c\mu\sum_{i=1}^{p}\phi_i + E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}^2\right] + 2E\left[\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right\}\right] + \sum_{i=1}^{q}\theta_i^2 + E\left[\sigma_t^2\right] - \mu^2\end{pmatrix}}{\partial\beta_i} \\
&= \frac{\partial\begin{pmatrix} c^2 + 2c\mu\sum_{i=1}^{p}\phi_i + E\left[\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}^2\right] + 2E\left[\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right] \\ +2E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\right] + \sum_{i=1}^{q}\theta_i^2 + E\left[\sigma_t^2\right] - \mu^2\end{pmatrix}}{\partial\beta_i} \\
&= 2E\left[\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]\frac{\partial\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}}{\partial\beta_i} + 2\frac{\partial E\left[\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]}{\partial\beta_i} \\
&\quad +2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\right]}{\partial\beta_i} \\
&= -2E\left[\sum_{i=1}^{p}\phi_i\varepsilon_{t-i}\right]\left(\phi_i X_{t-i}^{'}\right) + 2\frac{\partial E\left[\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\left\{\sum_{i=1}^{q}\theta_i\eta_{t-i}\right\}\right]}{\partial\beta_i} \\
&\quad +2\frac{\partial E\left[\sigma_t\eta_t\left\{\sum_{i=1}^{p}\phi_i\left(Y_{t-i} - X_{t-i}^{'}\beta\right)\right\}\right]}{\partial\beta_i}.
\end{aligned}$$

The following derivative is also necessary for the computation of (95). The first order derivative of the conditional mean, μ with respect to the variance parameters in δ, we then get the following

$$\begin{aligned}\frac{\partial\mu}{\partial\beta} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\left(\beta_1,\beta_2,\ldots,\beta_k\right)} \\ &= \begin{bmatrix}\frac{\partial\mu}{\partial\beta_1} \\ \frac{\partial\mu}{\partial\beta_2} \\ \vdots \\ \vdots \\ \frac{\partial\mu}{\partial\beta_{k-1}} \\ \frac{\partial\mu}{\partial\beta_k}\end{bmatrix}.\end{aligned} \tag{97}$$

where the i-th element of the vector in (97) is given by

$$\begin{aligned}\frac{\partial\mu}{\partial\beta_i} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\beta_i} \\ &= 0. \quad \forall\,\beta_i\end{aligned}$$

2) Second order derivatives The second order derivatives of the log-likelihood with respect to the vector of parameters, ξ, is also referred to as the Hessian. The Hessian is computed as follows:

$$H(\xi) = \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \xi \partial \xi'} \tag{98}$$

Since $\xi = (\beta', \delta')$ is of the dimensions $((k+2+p+q+m+n) \times 1)$, the Hessian will be computed in the following way. Given the first element of gradient, 90, the first row of the Hessian is derived by computing the derivative with respect the parameter vector as follows

$$\begin{aligned}\frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial \beta_1}{\partial \xi'} &= \begin{bmatrix} \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial \beta_1}{\partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial \beta_1}{\partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial \beta_1}{\partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial \beta_1}{\partial \gamma_m} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \gamma_m} \end{bmatrix}\end{aligned} \tag{99}$$

Repeating the process in (99) for all the elements of the gradient, (90) gives the Hessian as follows:

$$\begin{aligned} H(\xi) &= \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \xi \partial \xi'} \\ &= \begin{bmatrix} \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_1 \partial \gamma_m} \\ \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_2 \partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_2 \partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \beta_2 \partial \gamma_m} \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_{m-1} \partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_{m-1} \partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_{m-1} \partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_{m-1} \partial \gamma_m} \\ \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_m \partial \beta_1} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_m \partial \beta_2} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_m \partial \gamma_{m-1}} & \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)^*}{\partial \gamma_m \partial \gamma_m} \end{bmatrix} \end{aligned} \tag{100}$$

The Hessian matrix in this case will be equivalent to the information matrix which is an inverse of the variance-covariance matrix. The information matrix is computed by obtaining the second order derivatives with respect to $\xi = (\beta', \delta')$ as follows

$$\mathcal{I}_{\mathcal{T}}(\xi) = \begin{bmatrix} \mathcal{I}_{\delta\delta} & \mathcal{I}_{\delta\beta} \\ \mathcal{I}_{\beta\delta} & \mathcal{I}_{\beta\beta} \end{bmatrix} \tag{101}$$

Therefore the information matrix with respect to the regression coefficient parameter, β is found by computing the second order derivative of the log-likelihood with respect to $\beta = (\beta_1, \beta_2, \ldots, \beta_k)^{\backslash}$ which is essentially differentiating

(95) with respect to β. This is shown as follows

$$\begin{aligned}
\mathcal{I}_{\beta\beta} &= \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)}{\partial\beta\partial\beta'} \\
&= \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial\beta}{\partial\beta'} \\
&= \frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial(h_t^2)}{\partial\beta}\left\{1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}+\frac{\partial\mu}{\partial\beta}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\right)}{\partial\beta'} \\
&= \left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right)\frac{\partial\left(1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right)}{\partial\beta'} \\
&\quad+\left\{1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}\frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial(h_t^2)}{\partial\beta}\right)}{\partial\beta'}+\frac{\partial\mu}{\partial\beta}\frac{\partial\left(\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right)}{\partial\beta'} \\
&\quad+\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\frac{\partial\mu}{\partial\beta\partial\beta'}. \qquad (102)
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial\left(1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right)}{\partial\beta'} &= -\left\{\left[\left(Y_t-X_t'\beta\right)^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{1}{h_t^2}(2)\left(Y_t-X_t'\beta\right)\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\beta}\right]\right\} \\
&\quad+\left\{\left[(2)\left(Y_t-X_t'\beta\right)(\mu)(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{2\mu}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\beta}\right]\right\} \\
&\quad+\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]-\left\{\left[2\frac{\mu}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]+\left[\mu^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\right\} \\
&= \left[\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]-\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\beta}\right] \\
&\quad-\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\mu}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{2\mu}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\beta}\right] \\
&\quad+\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]-\left[\frac{2\mu}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]+\left[\frac{\mu^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right] \\
&= \frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}-\frac{2\left(Y_t-X_t'\beta\right)\mu}{h_t^2}+\frac{\mu^2}{h_t^2}\right\} \\
&\quad+\frac{\partial\mu}{\partial\beta}\left\{\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{2\mu}{h_t^2}\right\}+\frac{2}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\beta}\left\{\mu-\left(Y_t-X_t'\beta\right)\right\} \\
&= \frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}-\frac{2\left(Y_t-X_t'\beta\right)\mu}{h_t^2}+\frac{\mu^2}{h_t^2}\right\} \\
&\quad-\frac{2X_t'}{h_t^2}\left\{\mu-\left(Y_t-X_t'\beta\right)\right\}. \qquad (103)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right)}{\partial\beta'} &= \left[-\frac{1}{2}\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\right]+\left[-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\right] \\
&= \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'} \\
&= \frac{1}{2}\frac{1}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}-\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\right\}. \qquad (104)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\left(\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right)}{\partial\beta'} &= \left\{\left[\left(Y_t-X_t^{'}\beta\right)\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{1}{h_t^2}\frac{\partial\left(Y_t-X_t^{'}\beta\right)}{\partial\beta}\right]\right\} \\
&\quad -\left\{\left[\mu\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{1}{h_t^2}\frac{\partial\mu}{\partial\beta}\right]\right\} \\
&= \left\{-\left[\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]+\left[\frac{1}{h_t^2}\frac{\partial\left(Y_t-X_t^{'}\beta\right)}{\partial\beta}\right]\right\}+\left\{\left[\frac{\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right]\right\} \\
&= -\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}-\frac{X_t^{'}}{h_t^2}. \qquad (105)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\mu}{\partial\beta\partial\beta'} &= \frac{\partial\mu/\partial\beta}{\partial\beta'} \\
&= \frac{\partial 0}{\partial\beta'} \\
&= 0. \qquad (106)
\end{aligned}$$

It therefore follows that

$$\begin{aligned}
\mathcal{I}_{\beta\beta} &= \left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right)\left(\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}-\frac{2\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}+\frac{\mu^2}{h_t^2}\right\}+\left(\frac{2}{h_t^2}\right)\left(-X_t^{'}\right)\left\{\mu-\left(Y_t-X_t^{'}\beta\right)\right\}\right) \\
&+\left(1-\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right)\left(\frac{1}{2}\frac{1}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}-\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\right\}\right) \\
&+\left(\frac{\partial\mu}{\partial\beta}\right)\left(-\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}+\frac{1}{h_t^2}\left(-X_t^{'}\right)\right)+\left(\frac{\left(Y_t-X_t^{'}\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right)(0) \\
&= \left(-\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\right)\left(\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}-\frac{2\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}+\frac{\mu^2}{h_t^2}\right) \\
&+\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\right)\left(-\frac{2X_t^{'}}{h_t^2}\right)\left(-X_t^{'}\right)\left\{\mu-\left(Y_t-X_t^{'}\beta\right)\right\} \\
&+\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\left(1-\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right) \\
&-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\left(1-\frac{\left(Y_t-X_t^{'}\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t^{'}\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right) \\
&= -\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2-\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\left(Y_t-X_t^{'}\beta\right)X_t^{'}-X_t^{'}\mu\right\} \\
&+\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\left(1-\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right)-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\left(1-\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right) \\
&= \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\frac{\partial\left(h_t^2\right)}{\partial\beta'}\left(1-\frac{2}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right)-\frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left\{\left(Y_t-X_t^{'}\beta\right)X_t^{'}-X_t^{'}\mu\right\} \\
&-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\left(1-\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right) \\
&= \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta}\left[\frac{\partial\left(h_t^2\right)}{\partial\beta'}\left(1-\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right)-\left\{\left(Y_t-X_t^{'}\beta\right)X_t^{'}-X_t^{'}\mu\right\}\right] \\
&-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\beta\partial\beta'}\left(1-\frac{1}{h_t^2}\left\{\left(Y_t-X_t^{'}\beta\right)-\mu\right\}^2\right). \qquad (107)
\end{aligned}$$

It then follows that the information matrix with respect to the variance parameters, δ is found by computing the second order derivative of the log-likelihood with respect to $\delta=(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q,\,\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m)^{'}$

which is essentially differentiating (91) with respect to δ. This is shown as follows

$$\begin{aligned}
\mathcal{I}_{\delta\delta} &= \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)}{\partial\delta\partial\delta'} \\
&= \frac{\partial^2 \mathcal{L}_{\mathcal{T}}(\xi)/\partial\delta}{\partial\delta'} \\
&= \frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\left\{1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}+\frac{\partial\mu}{\partial\delta}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\right)}{\partial\delta'} \\
&= \left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right)\frac{\partial\left(1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right)}{\partial\delta'} \\
&\quad+\left\{1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}\frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right)}{\partial\delta'}+\frac{\partial\mu}{\partial\delta}\frac{\partial\left(\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right)}{\partial\delta'} \\
&\quad+\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\frac{\partial\mu}{\partial\delta\partial\delta'}. \qquad (108)
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial\left(1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right)}{\partial\delta'} &= 0-\left[\left(Y_t-X_t'\beta\right)^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]-\left[\frac{1}{h_t^2}(2)\left(Y_t-X_t'\beta\right)\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right] \\
&\quad+\left[(2)\left(Y_t-X_t'\beta\right)\mu(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[(2)\frac{\mu}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right] \\
&\quad+\left[(2)\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]-\left[(2)\mu\frac{1}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]-\left[\mu^2(-1)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right] \\
&= -\left[\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]-\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right] \\
&\quad-\left[\frac{2\left(Y_t-X_t'\beta\right)\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[\frac{2\mu}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right] \\
&\quad+\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]-\left[\frac{2\mu}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]+\left[\frac{\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right] \\
&= -\left[\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]-\left[\frac{2\left(Y_t-X_t'\beta\right)\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right] \\
&\quad+\left[\frac{2\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]-\left[\frac{2\mu}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]+\left[\frac{\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right] \\
&= -\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\} \\
&\quad+2\frac{\partial\mu}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}. \qquad (109)
\end{aligned}$$

where the derivatives of $\frac{\partial\left(h_t^2\right)}{\partial\delta'}$ and $\frac{\partial\mu}{\partial\delta'}$ are similar to those found in (93) and (94) respectively. We then also compute the derivative of the following

$$\begin{aligned}
\frac{\partial\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right)}{\partial\delta'} &= \left[-\frac{1}{2}\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right] \\
&= \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'} \\
&= \frac{1}{2}\frac{1}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\}. \qquad (110)
\end{aligned}$$

where $\frac{\partial\left(h_t^2\right)}{\partial\delta}=\frac{\partial\left(h_t^2\right)}{\partial\delta'}$, previously derived in (93) and the matrix

$$\begin{aligned}
\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'} &= \frac{\partial\left(h_t^2\right)/\partial\delta}{\partial\delta'} \\
&= \begin{bmatrix}
\frac{\partial^2\left(h_t^2\right)}{\partial c\partial c} & \frac{\partial^2\left(h_t^2\right)}{\partial c\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\left(h_t^2\right)}{\partial c\partial\gamma_{m-1}} & \frac{\partial^2\left(h_t^2\right)}{\partial c\partial\gamma_m} \\
\frac{\partial^2\mu}{\partial\phi_1\partial c} & \frac{\partial^2\mu}{\partial\phi_1\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\mu}{\partial\phi_1\partial\gamma_{m-1}} & \frac{\partial^2\mu}{\partial\phi_1\partial\gamma_m} \\
\vdots & \vdots & \ddots & & \vdots & \vdots \\
\vdots & \vdots & & \ddots & \vdots & \vdots \\
\frac{\partial^2\left(h_t^2\right)}{\partial\gamma_{m-1}\partial c} & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_{m-1}\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_{m-1}\partial\gamma_{m-1}} & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_{m-1}\partial\gamma_m} \\
\frac{\partial^2\left(h_t^2\right)}{\partial\gamma_m\partial c} & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_m\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_m\partial\gamma_{m-1}} & \frac{\partial^2\left(h_t^2\right)}{\partial\gamma_m\partial\gamma_m}
\end{bmatrix}
\end{aligned}$$

The following derivative is also required to compute (108), and is derived as follows

$$\begin{aligned}
\frac{\partial\left(\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right)}{\partial\delta'} &= \left\{\left[\left(Y_t-X_t'\beta\right)\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[\frac{1}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right]\right\} \\
&\quad -\left\{\left[\mu\left(-1\right)\left(h_t^2\right)^{-2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[\frac{1}{h_t^2}\frac{\partial\mu}{\partial\delta'}\right]\right\} \\
&= \left\{-\left[\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]+\left[\frac{1}{h_t^2}\frac{\partial\left(Y_t-X_t'\beta\right)}{\partial\delta'}\right]\right\}+\left\{\left[\frac{\mu}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\right]\right\} \\
&= -\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}+\frac{1}{h_t^2}\left(0\right) \\
&= -\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}. \qquad (111)
\end{aligned}$$

Lastly, we compute the last derivative necessary in the computation of (108) given by

$$\frac{\partial\mu}{\partial\delta\partial\delta'} = \frac{\partial\mu/\partial\delta}{\partial\delta'} \qquad (112)$$

where

$$\begin{aligned}\frac{\partial\mu}{\partial\delta} &= \frac{\partial\left(\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}\right)}{\partial\left(c,\,\phi_1,\,\phi_2,\ldots,\,\phi_p\,\theta_1,\,\theta_2,\ldots,\,\theta_q,\,\omega,\,\alpha_1,\,\alpha_2,\ldots,\,\alpha_n\,\gamma_1,\,\gamma_2,\ldots,\,\gamma_m\right)} \\ &= \begin{bmatrix} \frac{\partial\mu}{\partial c} \\ \frac{\partial\mu}{\partial\phi_1} \\ \vdots \\ \vdots \\ \frac{\partial\mu}{\partial\gamma_{m-1}} \\ \frac{\partial\mu}{\partial\gamma_m} \end{bmatrix}.\end{aligned}$$

which was derived in (94). This second order derivative in (112) will therefore produce a matrix of the following form

$$\frac{\partial\mu/\partial\delta}{\partial\delta'} = \begin{bmatrix} \frac{\partial^2\mu}{\partial c\partial c} & \frac{\partial^2\mu}{\partial c\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\mu}{\partial c\partial\gamma_{m-1}} & \frac{\partial^2\mu}{\partial c\partial\gamma_m} \\ \frac{\partial^2\mu}{\partial\phi_1\partial c} & \frac{\partial^2\mu}{\partial\phi_1\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\mu}{\partial\phi_1\partial\gamma_{m-1}} & \frac{\partial^2\mu}{\partial\phi_1\partial\gamma_m} \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ \frac{\partial^2\mu}{\partial\gamma_{m-1}\partial c} & \frac{\partial^2\mu}{\partial\gamma_{m-1}\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\mu}{\partial\gamma_{m-1}\partial\gamma_{m-1}} & \frac{\partial^2\mu}{\partial\gamma_{m-1}\partial\gamma_m} \\ \frac{\partial^2\mu}{\partial\gamma_m\partial c} & \frac{\partial^2\mu}{\partial\gamma_m\partial\phi_1} & \cdots & \cdots & \frac{\partial^2\mu}{\partial\gamma_m\partial\gamma_{m-1}} & \frac{\partial^2\mu}{\partial\gamma_m\partial\gamma_m} \end{bmatrix} \tag{113}$$

of the dimensions $((2+p+q+m+n)\times(2+p+q+m+n))$. An explicit form of (113) would be extensive and hence will be left in this form at this time.

Finally we can substitute (109)e(113) into (108) and this gives

$$
\begin{aligned}
\mathcal{I}_{\delta\delta} = {} & \left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right)\left[-\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}\right] \\
& +\left(-\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\right)\left[2\frac{\partial\mu}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\right] \\
& +\left[1-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right]\left[\frac{1}{2}\frac{1}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\}\right] \\
& +\frac{\partial\mu}{\partial\delta}\left[-\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\right]+\left[\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right]\left[\frac{\partial\mu}{\partial\delta\partial\delta'}\right] \\
= {} & \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\}-\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\mu}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\} \\
& +\frac{1}{2}\frac{1}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\}-\frac{1}{2}\frac{1}{h_t^2}\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\} \\
& +\frac{1}{h_t^2}\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\}-\frac{1}{2}\frac{1}{h_t^2}\frac{\mu^2}{h_t^2}\left\{\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}-\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\right\} \\
& -\frac{1}{h_t^2}\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}+\frac{1}{h_t^2}\frac{\mu}{h_t^2}\frac{\partial\mu}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}+\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}\frac{\partial\mu}{\partial\delta\partial\delta'}-\frac{\mu}{h_t^2}\frac{\partial\mu}{\partial\delta\partial\delta'} \\
= {} & \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{1+\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}-\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}+2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\} \\
& -\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\left\{1+\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}-2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\} \\
& -\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\mu}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}+\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}+\frac{\partial\mu}{\partial\delta\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\} \\
= {} & \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{1+\frac{4\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{2\mu^2}{h_t^2}\right\} \\
& -\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\left\{1+\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}-2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\} \\
& -\frac{2}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\mu}{\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}+\frac{\partial\mu}{\partial\delta\partial\delta'}\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\} \\
= {} & \frac{1}{2}\frac{1}{h_t^2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\left(h_t^2\right)}{\partial\delta'}\left\{1+\frac{4\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{2\mu^2}{h_t^2}\right\} \\
& -\frac{1}{2}\frac{1}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta\partial\delta'}\left\{1+\frac{\left(Y_t-X_t'\beta\right)^2}{h_t^2}-2\frac{\left(Y_t-X_t'\beta\right)\mu}{h_t^2}-\frac{\mu^2}{h_t^2}\right\} \\
& +\left\{\frac{\left(Y_t-X_t'\beta\right)}{h_t^2}-\frac{\mu}{h_t^2}\right\}\left\{\frac{\partial\mu}{\partial\delta\partial\delta'}-\frac{2}{h_t^2}\frac{\partial\left(h_t^2\right)}{\partial\delta}\frac{\partial\mu}{\partial\delta'}\right\}. \qquad (114)
\end{aligned}
$$

Appendix B.

```
#***********************************************#
# This is the relevant R code for the analysis of #
# Bitcoin data in Chapter 3.   #
#***********************************************#

###################################################
### code chunk number 1:
###################################################
# Set working directory.
setwd("C:/Users/zola/Documents/Zola/Research/Code")

###################################################
### code chunk number 2:
###################################################
# Install and load relevant packages for our analysis.

install.packages("coinmarketcapr")
install.packages("crypto")
install.packages("tseries")
install.packages("ggplot2")
install.packages("fBasics")
install.packages("normtest")
install.packages("xts")
install.packages("reshape2")
install.packages("quantmod")
install.packages("rugarch")
install.packages("fGarch")



library(coinmarketcapr)
library(crypto)
library(tseries)
library(ggplot2)
library(fBasics)
library(normtest)
library(xts)
library(reshape2)
library(quantmod)
library(rugarch)
library(fGarch)

###################################################
### code chunk number 3:
###################################################
# Extract Bitcoin price data between 31 March 2016 to 31 March 2018.

BTC = crypto_history("bitcoin", start_date = "20160331", end_date = "20180331")
head(BTC)

# The descriptive statistics of Bitcoin closing price data.
basicStats(BTC$close, ci = 0.95)

# Plot the time series of the Bitcoin closing price data.
# Define the time series with dates and then plot the data.
dates = seq(as.Date("2016-03-31"), length = length(BTC$close), by = "days")
BTCts = xts(x = BTC$close, order.by = dates)
plot(BTCts, type = "l", ylab = "Price", xlab = "Date", main = " ")

# Plot the model residuals.
par(mfrow = c(1,1))
BTCts_df = data.frame(BTC$close, dates)
#BTCts_df$dates <- as.Date(dates)
require(scales)
ggplot(BTCts_df, aes(x = dates, y = BTC$close)) +
geom_line(color = "#0B0B0B", size = 0.5) + xlab("Date, t") +ylab("Price, US$ "~(P[t]))+
geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1)+
scale_x_date(labels = date_format("%b %d %Y"), breaks = date_breaks("2 month"))+
scale_y_continuous(limits = c(0, 20000), breaks = c(0, 2000, 4000, 6000, 8000, 10000, 12000, 14000, 16000,
theme_bw()+
ggsave("time_plot_BTC.png", height = 4, width = 5)

# Decomposition of multiplicative time series.
tsBTC = ts(BTC$close,  frequency = 365, start = c(2016, 90), end = c(2018, 90))
decomposedRes = decompose(tsBTC, type = "mult")
```

```
plot(decomposedRes)

# Testing for stationarity of the Bitcoin closing price data.
# 1. Autocorrelation and partial autocorrelation functions.
acf(BTCts)
pacf(BTCts)

# 2. Augmented Dickey-Fuller Test for non-staionarity.
adf.test(BTCts)

###################################################
### code chunk number 4:
###################################################
# Defining the log return time series data.

# Daily Exchange Rate of Bitcoin to USD from 2016 to 2018
log_BTC = log(BTC$close)
log_rBTC = diff(log(BTC$close), lag = 1)

# Time series of Bitcoin log returns.
rBTC = log(BTCts/lag(BTCts, -1))
summary(rBTC)

# Converting the price data to log returns may produce NA values.
# Removal of the last missing value (i.e. NA values).
rBTC = na.omit(rBTC)
summary(rBTC)

# Strip off dates and create a numeric object.
log_returnsBTC = coredata(rBTC)

# Descriptive statistics of Bitcoin log returns.
basicStats(rBTC, ci = 0.95)

# Plot the log returns price series - indicates an upward trend.
plot.ts(log_rBTC, ylab = expression(log(P[t])),  xlab = "Time Index", lwd = 1)
abline(h = 0, lty = 2)
```

```
# Plot the time series log returns data.
plot(rBTC, ylab = expression(Y[t]),  xlab = "Time Index", lwd = 1)
abline(h = 0, lty = 2)

# Define the time series with dates and then plot the data.
dates1 = seq(as.Date("2016-03-31"), length = length(rBTC), by = "days")

# Plot the time series log returns data.
par(mfrow = c(1,1))
rBTCts_df = data.frame(rBTC, dates1)
rBTCts_df$dates1 <- as.Date(dates1)
require(scales)
ggplot(rBTCts_df, aes(x = dates1, y = rBTC)) +
theme_bw()+
geom_line(color = "#0B0B0B", size = 0.5) + xlab("Date, t") +ylab(expression(Y[t]))+
geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1)+
scale_x_date(labels = date_format("%b %d %Y"), breaks = date_breaks("4 month"))+
scale_y_continuous(limits = c(-0.3, 0.3), breaks = c(-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3))+
ggsave("time_plot_rBTC.png", height = 4, width = 5)

# Testing for stationarity of the Bitcoin log returns closing price data.
# 1. Autocorrelation and partial autocorrelation functions.
acf(rBTC)
pacf(rBTC)

# 2. Augmented Dickey-Fuller Test for non-staionarity
adf.test(rBTC)

###################################################
### code chunk number 5:
###################################################
# Testing for the underlying distribution of the log returns,
# and testing for normality.

# 2-sided t-test.
t.test(log_returnsBTC, mu = 0, alt = "two.sided", conf = 0.95)

# Skewness test under the hypothesis of normality.
```

```
skewness.norm.test(log_returnsBTC)

# Kurtosis test under the hypothesis of normality.
kurtosis.norm.test(log_returnsBTC)

# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(log_returnsBTC)

# Histogram of log returns distribution.
hist(rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,20))
lines(density(rBTC), lwd = 2, col = "red")

# Q-Q plot of log returns distribution.
qqnorm(rBTC, pch = 19)
qqline(rBTC, col = "red", lwd = 2)

###################################################
### code chunk number 6:
###################################################
# Fitting parametric distributions to the log returns,
# to determine the underlying distibution of the data.

# Install and load relevant packages for our analysis.
install.packages("QRM")
install.packages("ghyp")

library(QRM)
library(ghyp)

###################################################################
# Fit a normal distribution to the Bitcoin log returns.
normfit = fit.norm(rBTC)
attributes(normfit)

#Define the parameters of the fitted distribution.
nmu = normfit$mu
nsigma = normfit$Sigma
```

```
normpars = c(nmu, nsigma)

#compute the AIC of the fitted distribution.
AIC_normfit = (-2*normfit$ll.max)+(2*length(normpars))
AIC_normfit

#log_likelihood = 1250.545
#AIC =  -2497.09

# Plot a histogram of the Bitcoin log returns with the fitted
# normal distribution.
xfit = sort(log_rBTC)
yfit = dnorm(xfit, mean = mean(rBTC), sd = sd(rBTC))
xvals = sort(log_rBTC)
yvals = dnorm(xvals, mean = nmu, sd = nsigma)
hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,40))

# Fitting a distribution curve with the estimated values from the fit.
lines(xvals, yvals, lwd = 2, col = "darkblue")

# Fitting a distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Note:
# It is found that the true distribution is a much heavier tailed
# than the normal.

###################################################################
# Fit a Student-t distribution to the Bitcoin log returns.
tfit = fit.st(log_rBTC)

# Define the parameters of the fitted distribution.
tpars = tfit$par.ests
tnu = tpars[1]
tmu = tpars[2]
tsigma = tpars[3]
```

```
# Compute the AIC of the fitted distribution.
AIC_tfit = (-2*tfit$ll.max)+(2*length(tpars))
AIC_tfit

#log_likelihood = 1350.865
#AIC = -2695.73

# Plot a histogram of the Bitcoin log returns with the fitted
# Student-t distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,20))

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted density from the parameter estimates of the log_rBTC.
xvals = sort(log_rBTC)
yvals = dt((xvals - tmu)/tsigma, df = tnu)/tsigma

# Superimpose a blue line to show the fitted t density
lines(xvals, yvals, lwd = 2, col = "blue")

####################################################################
# Fit a GED distribution to the Bitcoin log returns.
gedfit = .gedFit(log_rBTC)

# Define the parameters of the fitted distribution.
gedpars = gedfit@fit$'estimate'
gedmu = gedpars[1]
gedsigma = gedpars[2]
gednu = gedpars[3]

# Compute the AIC of the fitted distribution.
AIC_gedfit = (-2*gedfit@fit$minimum)+(2*length(gedpars))
AIC_gedfit
```

```
#log_likelihood = 1377.069
#AIC =  -2748.139

# Plot a histogram of the Bitcoin log returns with the fitted
# GED distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
     ylim = c(0,20))

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted GED density from the parameter estimates of the log_rBTC.
xvals = sort(log_rBTC)
yvals = dged(xvals, mean = gedmu, sd = gedsigma, nu = gednu)

# Superimpose lines to show the fitted GED density.
lines(xvals, yvals, lwd = 2, col = "blue")

###################################################################
# Fit a symmetric hyperbolic distribution to the Bitcoin log returns.
hypfit = fit.NH(log_rBTC, case = "HYP", symmetric = TRUE)

# Define the parameters of the fitted distribution.
hypaltpars = hypfit$alt.pars
hypd = hypaltpars[1]
hypa = hypaltpars[2]
hypb = hypaltpars[3]
hypmu = hypaltpars[4]
hyplambda = 1
hypgamma = 0

theta = c(hyplambda, hypa, hypb, hypd, hypmu)

# Compute the AIC of the fitted distribution.
```

```
AIC_hypfit = (-2*hypfit$ll.max)+(2*length(hyppars))

AIC_hypfit


#log_likelihood = 1273.354

#AIC = -2540.709


# Plot a histogram of the Bitcoin log returns with the fitted

# symmetric hyperbolic distribution.

xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))

yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))


hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),

 ylim = c(0,20))


# Fitting a normal distribution curve with the actual true values of the data.

lines(xfit, yfit, lwd = 2, col = "red")


# Compute the fitted symmetric hyperbolic density from the parameter estimates

# of the log_rBTC.

xvals = sort(log_rBTC)

yvals = dghypB(xvals, lambda = hyplambda, delta = hypd, alpha = hypa,

   beta = hypb, mu = hypmu)


# Superimpose lines to show the fitted symmetric hyperbolic density.

lines(xvals, yvals, lwd = 2, col = "green")


###################################################################

# Fit a skew hyperbolic distribution to the Bitcoin log returns.

hypfit2 = fit.NH(log_rBTC, case = "HYP", symmetric = FALSE)


# Define the parameters of the fitted distribution.

hyppars2 = hypfit2$par.ests

shypchi = hyppars2[1]

shyppsi = hyppars2[2]

shypmu = hyppars2[3]

shypgamma = hyppars2[4]


hypaltpars2 = hypfit2$alt.pars
```

```
shypd = hypaltpars2[1]
shypa = hypaltpars2[2]
shypb = hypaltpars2[3]
shypmu = hypaltpars2[4]
shyplambda = 1

# Compute the AIC of the fitted distribution.
AIC_shypfit = (-2*hypfit2$ll.max)+(2*length(hyppars))
AIC_shypfit

AIC_shypfit2 = (-2*hypfit2$ll.max)+(2*length(hypaltpars2))
AIC_shypfit2

#log_likelihood = 1272.241
#AIC = -2538.481
#AIC2 = -2536.481

# Plot a histogram of the Bitcoin log returns with the fitted
# skew hyperbolic distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,20))

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted symmetric hyperbolic density from the parameter estimates
# of the log_rBTC.
xvals = sort(log_rBTC)
yvals = dghypB(xvals, lambda = shyplambda, delta = shypd, alpha = shypa,
   beta = shypb, mu = shypmu)

# Superimpose lines to show the fitted skew hyperbolic density.
lines(xvals, yvals, lwd = 2, col = "green")

###################################################################
```

```
# Fit a symmetric normal inverse gaussian (nig) distribution to the Bitcoin
# log returns.
nigfit = fit.NH(log_rBTC, case = "NIG", symmetric = TRUE)
attributes(nigfit)

# Define the parameters of the fitted distribution.
nigpars = nigfit$par.ests
nigchi = nigpars[1]
nigpsi = nigpars[2]
nigmu = nigpars[3]

nigaltpars = nigfit$alt.pars
nigd = nigaltpars[1]
niga = nigaltpars[2]
nigb = nigaltpars[3]
nigmu = nigaltpars[4]
niglambda = -1/2
niggamma = sqrt((niga^2)-(nigb^2))

# Compute the AIC of the fitted distribution.
AIC_nigfit = (-2*nigfit$ll.max)+(2*length(nigpars))
AIC_nigfit

AIC_nigfit2 = (-2*nigfit$ll.max)+(2*length(nigaltpars))
AIC_nigfit2

#log_likelihood = 1363.009
#AIC = -2720.018
#AIC2 = -2718.018

# Plot a histogram of the Bitcoin log returns with the fitted
# symmetric NIG distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,20))
```

```
# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted NIG density from the parameter estimates of the log_rBTC.
xvals = sort(log_rBTC)
yvals = dghyp(xvals, lambda = niglambda, chi = nigchi, psi = nigpsi,
  mu = nigmu, gamma = niggamma)
yvals2 = dghypB(xvals, lambda = niglambda, delta = nigd, alpha = niga,
beta = nigb, mu = nigmu)

# Superimpose lines to show the fitted NIG density.
lines(xvals, yvals, lwd = 2, col = "blue")
lines(xvals, yvals2, lwd = 2, col = "green")

###################################################################
# Fit a skew normal inverse gaussian (NIG) distribution to the Bitcoin log
# returns.
nigfit2 = fit.NH(log_rBTC, case = "NIG", symmetric = FALSE)
attributes(nigfit2)

# Define the parameters of the fitted distribution.
nigpars2 = nigfit2$par.ests
snigchi = nigpars2[1]
snigpsi = nigpars2[2]
snigmu = nigpars2[3]
sniggamma = nigpars2[4]

nigaltpars2 = nigfit2$alt.pars
snigd = nigaltpars2[1]
sniga = nigaltpars2[2]
snigb = nigaltpars2[3]
snigmu = nigaltpars2[4]
sniglambda = -1/2

# Compute the AIC of the fitted distribution.
AIC_snigfit = (-2*nigfit2$ll.max)+(2*length(nigpars2))
AIC_snigfit
```

```
AIC_snigfit2 = (-2*nigfit2$ll.max)+(2*length(nigaltpars2))
AIC_snigfit2

#log_likelihood = 1363.116
#AIC =  -2718.231
#AIC2 = -2718.231

# Plot a histogram of the Bitcoin log returns with the fitted
# skewed NIG distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,20))

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted symmetric hyperbolic density from the parameter estimates of
# the log_rBTC.
xvals = sort(log_rBTC)
yvals = dghyp(xvals, lambda = sniglambda, chi = snigchi, psi = snigpsi,
  mu = snigmu, gamma = sniggamma)
yvals2 = dghypB(xvals, lambda = sniglambda, delta = snigd, alpha = sniga,
beta = snigb, mu = snigmu)

# Superimpose lines to show the fitted symmetric hyperbolic density.
lines(xvals, yvals, lwd = 2, col = "blue")
lines(xvals, yvals2, lwd = 2, col = "green")

######################################################################
# Fit a symmetric generalized hyperbolic (GHYP) distribution to the Bitcoin
# log returns.
ghypfit = fit.ghypuv(log_rBTC, symmetric = TRUE)

# Define the parameters of the fitted distribution.
ghyppars = ghypfit@fitted.params
ghypchi = ghypfit@chi
```

```
ghyppsi = ghypfit@psi

ghypmu = ghypfit@mu

ghypsigma = ghypfit@sigma

ghypa_bar = ghypfit@alpha.bar

ghyplambda = ghypfit@lambda

ghypgamma = ghypfit@gamma



# Compute the log-likelihood and AIC of the fitted distribution.

Loglik_ghypfit = ghypfit@llh

Loglik_ghypfit


AIC_ghypfit = ghypfit@aic

AIC_ghypfit


#log_likelihood =  1380.43

#AIC = -2752.86


# Plot a histogram of the Bitcoin log returns with the fitted

# symmetric GHYP distribution and normal distribution.

xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))

yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))


hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),

 ylim = c(0,25))


# Fitting a normal distribution curve with the actual true values of the data.

lines(xfit, yfit, lwd = 2, col = "red")


# Compute the fitted symmetric generalized hyperbolic density from the parameter

# estimates of the log_rBTC.

xvals = sort(log_rBTC)

yvals = dghyp(xvals, object = ghyp(lambda = ghyplambda, alpha.bar = ghypa_bar,

  mu = ghypmu, sigma = ghypsigma, gamma = ghypgamma))


# Superimpose lines to show the fitted symmetric generalized hyperbolic density.

lines(xvals, yvals, lwd = 2, col = "blue")
```

```
###################################################################
# Fit a skewed generalized hyperbolic (GHYP) distribution to the Bitcoin log returns.
ghypfit2 = fit.ghypuv(log_rBTC, symmetric = FALSE)

# Define the parameters of the fitted distribution.
sghyppars = ghypfit2@fitted.params
sghypchi = ghypfit2@chi
sghyppsi = ghypfit2@psi
sghypmu = ghypfit2@mu
sghypsigma = ghypfit2@sigma
sghypa_bar = ghypfit2@alpha.bar
sghyplambda = ghypfit2@lambda
sghypgamma = ghypfit2@gamma



# Compute the log-likelihood and AIC of the fitted distribution.
Loglik_sghypfit = ghypfit2@llh
Loglik_sghypfit

AIC_sghypfit = ghypfit2@aic
AIC_sghypfit

#log_likelihood =  1380.531
#AIC =  -2751.062

# Plot a histogram of the Bitcoin log returns fitted
# skewed GHYP distribution and normal distribution.
xfit = seq(min(log_rBTC), max(log_rBTC), length = length(log_rBTC))
yfit = dnorm(xfit, mean = mean(log_rBTC), sd = sd(log_rBTC))

hist(log_rBTC, nclass = 20, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,30))

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit, yfit, lwd = 2, col = "red")

# Compute the fitted skewed generalized hyperbolic density from the parameter estimates
# of the log_rBTC.
```

```
xvals = sort(log_rBTC)
yvals = dghyp(xvals, object = ghyp(lambda = sghyplambda, alpha.bar = sghypa_bar,
  mu = sghypmu, sigma = sghypsigma, gamma = sghypgamma))

# Superimpose lines to show the fitted symmetric generalized hyperbolic density.
lines(xvals, yvals, lwd = 2, col = "blue")

#--------------------------------------------#
# Histogram and best model fitted densities.
#--------------------------------------------#
# Plot a histogram of the Bitcoin log returns.
hist(log_rBTC, nclass = 100, probability = TRUE, xlab = expression('Log returns'),
 ylim = c(0,25), main = " ")

# Compute the fitted symmetric GHYP density and GED density,
# from the parameter estimates of the log_rBTC.
xvals = sort(log_rBTC)
yvals = dghyp(xvals, object = ghyp(lambda = ghyplambda, alpha.bar = ghypa_bar,
  mu = ghypmu, sigma = ghypsigma, gamma = ghypgamma))
yvals2 = dged(xvals, mean = gedmu, sd = gedsigma, nu = gednu)

# Superimpose lines to show the fitted symmetric GHYP density.
lines(xvals, yvals, lwd = 2, col = "darkblue")

# Superimpose lines to show the fitted GED density.
lines(xvals, yvals2, lwd = 1, col = "red")



###################################################
### code chunk number 7:
###################################################
# Now we will fit an ARMA-GARCH process to the Bitcoin log returns data,
# with different conditional distributions.

# These models consider the normal distribution as the conditional distribution.

# Our 1st model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc11 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "norm")
model1.11 = ugarchfit(spec = btc11, data = rBTC)
model1.11

model1.11_fit_res2 = model1.11@fit$residuals
model1.11_fit_var = model1.11@fit$var

plot(model1.11_fit_res2, type = 'l')
lines(model1.11_fit_var, col = "red")

#----------------------------------------------------------------------------#

# Our 2nd model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc21 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "norm")
model1.21 = ugarchfit(spec = btc21, data = rBTC)
model1.21

model1.21_fit_res2 = model1.21@fit$residuals
model1.21_fit_var = model1.21@fit$var

plot(model1.21_fit_res2, type = 'l')
lines(model1.21_fit_var, col = "red")

#----------------------------------------------------------------------------#


# Our 3rd model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc31= ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
  mean.model = list(armaOrder = c(0,1)), distribution.model = "norm")
model1.31 = ugarchfit(spec = btc31, data = rBTC)
model1.31

model1.31_fit_res2 = model1.31@fit$residuals
model1.31_fit_var = model1.31@fit$var

plot(model1.31_fit_res2, type = 'l')
lines(model1.31_fit_var, col = "red")
```

```
#---------------------------------------------------------------------------#



# Our 4th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc41 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "norm")
model1.41 = ugarchfit(spec = btc41, data = rBTC)
model1.41

model1.41_fit_res2 = model1.41@fit$residuals
model1.41_fit_var = model1.41@fit$var

plot(model1.41_fit_res2, type = 'l')
lines(model1.41_fit_var, col = "red")

#---------------------------------------------------------------------------#

# Our 5th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc51 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "norm")
model1.51 = ugarchfit(spec = btc51, data = rBTC)
model1.51

model1.51_fit_res2 = model1.51@fit$residuals
model1.51_fit_var = model1.51@fit$var

plot(model1.51_fit_res2, type = 'l')
lines(model1.51_fit_var, col = "red")

#---------------------------------------------------------------------------#

# Our 6th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc61 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "norm")
model1.61 = ugarchfit(spec = btc61, data = rBTC)
model1.61
```

```
model1.61_fit_res2 = model1.61@fit$residuals
model1.61_fit_var = model1.61@fit$var

plot(model1.61_fit_res2, type = 'l')
lines(model1.61_fit_var, col = "red")



#-----------------------------------------------------------------------------#

# Our 7th model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc71 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "norm")
model1.71 = ugarchfit(spec = btc71, data = rBTC)
model1.71

model1.71_fit_res2 = model1.71@fit$residuals
model1.71_fit_var = model1.71@fit$var

plot(model1.71_fit_res2, type = 'l')
lines(model1.71_fit_var, col = "red")

####################################################################################
# These models consider a Student-t distribution as the conditional distribution.

# Our 8th model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc12 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "std")
model1.12 = ugarchfit(spec = btc12, data = rBTC)
model1.12

#-----------------------------------------------------------------------------#

# Our 9th model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc22 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "std")
model1.22 = ugarchfit(spec = btc22, data = rBTC)
model1.22
```

```
#---------------------------------------------------------------------------#



# Our 10th model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc32 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "std")
model1.32 = ugarchfit(spec = btc32, data = rBTC)
model1.32

#---------------------------------------------------------------------------#

# Our 11th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc42 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "std")
model1.42 = ugarchfit(spec = btc42, data = rBTC)
model1.42

#---------------------------------------------------------------------------#

# Our 12th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc52 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "std")
model1.52 = ugarchfit(spec = btc52, data = rBTC)
model1.52

#---------------------------------------------------------------------------#

# Our 13th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc62 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "std")
model1.62 = ugarchfit(spec = btc62, data = rBTC)
model1.62


#---------------------------------------------------------------------------#

# Our 14th model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc72 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "std")
model1.72 = ugarchfit(spec = btc72, data = rBTC)
model1.72



####################################################################################
# These models consider the generalized error distribution (GED) as the conditional
# distribution.

# Our 15th model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc13 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "ged")
model1.13 = ugarchfit(spec = btc13, data = rBTC)
model1.13

#----------------------------------------------------------------------------#

# Our 16th model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc23 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "ged")
model1.23 = ugarchfit(spec = btc23, data = rBTC)
model1.23

#----------------------------------------------------------------------------#

# Our 17th model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc33 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "ged")
model1.33 = ugarchfit(spec = btc33, data = rBTC)
model1.33

#----------------------------------------------------------------------------#

# Our 18th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc43 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "ged")
model1.43 = ugarchfit(spec = btc43, data = rBTC)
```

```
model1.43

#----------------------------------------------------------------------------#



# Our 19th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc53 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "ged")
model1.53 = ugarchfit(spec = btc53, data = rBTC)
model1.53

#----------------------------------------------------------------------------#

# Our 20th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc63 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "ged")
model1.63 = ugarchfit(spec = btc63, data = rBTC)
model1.63

#----------------------------------------------------------------------------#

# Our 21st model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc73 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "ged")
model1.73 = ugarchfit(spec = btc73, data = rBTC)
model1.73



##################################################################################
# These models consider the skew-generalized error distribution as the conditional
# ditribution.

# Our 22nd model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc14 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "sged")
model1.14 = ugarchfit(spec = btc14, data = rBTC)
model1.14
```

```
#--------------------------------------------------------------------------#

# Our 23rd model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc24 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "sged")
model1.24 = ugarchfit(spec = btc24, data = rBTC)
model1.24

#--------------------------------------------------------------------------#

# Our 24th model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc34 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "sged")
model1.34 = ugarchfit(spec = btc34, data = rBTC)
model1.34

#--------------------------------------------------------------------------#

# Our 25th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc44 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "sged")
model1.44 = ugarchfit(spec = btc44, data = rBTC)
model1.44

#--------------------------------------------------------------------------#

# Our 26th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc54 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "sged")
model1.54 = ugarchfit(spec = btc54, data = rBTC)
model1.54

#--------------------------------------------------------------------------#

# Our 27th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc64 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "sged")
model1.64 = ugarchfit(spec = btc64, data = rBTC)
model1.64

#----------------------------------------------------------------------------#

# Our 28th model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc74 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "sged")
model1.74 = ugarchfit(spec = btc74, data = rBTC)
model1.74

##################################################################################
# These models consider the generalized hyperbolic distribution as the conditional
# distribution.

# Our 29th model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc15 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "ghyp")
model1.15 = ugarchfit(spec = btc15, data = rBTC)
model1.15

#----------------------------------------------------------------------------#

# Our 30th model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc25 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "ghyp")
model1.25 = ugarchfit(spec = btc25, data = rBTC)
model1.25

#----------------------------------------------------------------------------#

# Our 31st model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc35 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "ghyp")
model1.35 = ugarchfit(spec = btc35, data = rBTC)
model1.35
```

```
#----------------------------------------------------------------------------#

# Our 32nd model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc45 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "ghyp")
model1.45 = ugarchfit(spec = btc45, data = rBTC)
model1.45

#----------------------------------------------------------------------------#

# Our 33rd model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc55 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "ghyp")
model1.55 = ugarchfit(spec = btc55, data = rBTC)
model1.55

#----------------------------------------------------------------------------#

# Our 34th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc65 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "ghyp")
model1.65 = ugarchfit(spec = btc65, data = rBTC)
model1.65

#----------------------------------------------------------------------------#

# Our 35th model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc75 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "ghyp")
model1.75 = ugarchfit(spec = btc75, data = rBTC)
model1.75

#----------------------------------------------------------------------------#
####################################################################################
# These models consider the normal inverse gaussian (NIG) distribution the conditional
# distribution.

# Our 36th model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
```

```
btc16 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "nig")
model1.16 = ugarchfit(spec = btc16, data = rBTC)
model1.16

#------------------------------------------------------------------------------#

# Our 37th model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc26 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model = "nig")
model1.26 = ugarchfit(spec = btc26, data = rBTC)
model1.26

#------------------------------------------------------------------------------#

# Our 38th model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc36 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "nig")
model1.36 = ugarchfit(spec = btc36, data = rBTC)
model1.36

#------------------------------------------------------------------------------#

# Our 39th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc46 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "nig")
model1.46 = ugarchfit(spec = btc46, data = rBTC)
model1.46

#------------------------------------------------------------------------------#

# Our 40th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc56 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "nig")
model1.56 = ugarchfit(spec = btc56, data = rBTC)
model1.56

#------------------------------------------------------------------------------#
```

```
# Our 41st model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc66 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "nig")
model1.66 = ugarchfit(spec = btc66, data = rBTC)
model1.66

#----------------------------------------------------------------------------#

# Our 42nd model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc76 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "nig")
model1.76 = ugarchfit(spec = btc76, data = rBTC)
model1.76

##################################################################################
##################################################################################
# These models consider a Johnson's SU distribution model.

# Our 43rd model fitted will have ARMA(0,0)-GARCH(1,1) parameters.
btc17 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,0)), distribution.model = "jsu")
model1.17 = ugarchfit(spec = btc17, data = rBTC)
model1.17

#----------------------------------------------------------------------------#

# Our 44th model fitted will have ARMA(1,0)-GARCH(1,1) parameters.
btc27 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,0)), distribution.model ="jsu")
model1.27 = ugarchfit(spec = btc27, data = rBTC)
model1.27

#----------------------------------------------------------------------------#

# Our 45th model fitted will have ARMA(0,1)-GARCH(1,1) parameters.
btc37 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(0,1)), distribution.model = "jsu")
```

```
model1.37 = ugarchfit(spec = btc37, data = rBTC)
model1.37

#----------------------------------------------------------------------------#

# Our 46th model fitted will have ARMA(1,1)-GARCH(1,1) parameters.
btc47 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,1)), distribution.model = "jsu")
model1.47 = ugarchfit(spec = btc47, data = rBTC)
model1.47

#----------------------------------------------------------------------------#

# Our 47th model fitted will have ARMA(2,1)-GARCH(1,1) parameters.
btc57 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,1)), distribution.model = "jsu")
model1.57 = ugarchfit(spec = btc57, data = rBTC)
model1.57

#----------------------------------------------------------------------------#

# Our 48th model fitted will have ARMA(1,2)-GARCH(1,1) parameters.
btc67 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(1,2)), distribution.model = "jsu")
model1.67 = ugarchfit(spec = btc67, data = rBTC)
model1.67


#----------------------------------------------------------------------------#

# Our 49th model fitted will have ARMA(2,2)-GARCH(1,1) parameters.
btc77 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
   mean.model = list(armaOrder = c(2,2)), distribution.model = "jsu")
model1.77 = ugarchfit(spec = btc77, data = rBTC)
model1.77


##################################################
```

```
### code chunk number 8:
###################################################
# Model forecating of the best fitted model.

# Plot forecasts for competing models.
# Model forecasting - using the ARMA(0,0)-GARCH(1,1) under "GED" parameter estimates.
model1.13_for = ugarchforecast(model1.13 , data = NULL, n.ahead = 10, n.roll = 0, out.sample = 0)

# View different plots of the forecast
#par(mfrow = c(1, 2))
plot(model1.13, which = "all")

# We are particularly interested in the sigma (volatility) forecasts
model1.13_forecast = model1.13_for@forecast$sigmaFor
plot(model1.13_forecast)

#Note:
# We can tell that we are forecasting an increase in volatility over a 10 day period.

#--------------------------------------------------------------------------------#

# Model forecasting - using the ARMA(2,2)-GARCH(1,1) under "GED" parameter estimates.
model1.73_for = ugarchforecast(model1.73 , data = NULL, n.ahead = 10, n.roll = 0, out.sample = 0)
model1.73_for

# View different plots of the forecast
#par(mfrow = c(1, 2))
plot(model1.73, which = "all")

# We are particularly interested in the sigma (volatility) forecasts
model1.73_forecast = model1.73_for@forecast$sigmaFor
plot(model1.73_forecast)

#Note:
# We can tell that we are forecasting an increase in volatility over a 10 day period.

#--------------------------------------------------------------------------------#
```

```
# Plot forecasts for competing models.
# Model forecasting - using the ARMA(0,0)-GARCH(1,1) under "GED" parameter estimates.
model1.13_frct = ugarchforecast(model1.13 , data = NULL, n.ahead = 10, n.roll = 10, out.sample = 0)

# Model forecasting - using the ARMA(2,2)-GARCH(1,1) under "GED" parameter estimates.
model1.73_frct = ugarchforecast(model1.73 ,data = NULL, n.ahead = 10, n.roll = 10, out.sample = 0)

#---------------------------------------------------------------------------#

# Extract volatility forecasts.
model1.13_frct.sigma = as.data.frame(model1.13_frct@'forecast'$sigmaFor)
model1.73_frct.sigma = as.data.frame(model1.73_frct@'forecast'$sigmaFor)

ymax = max(model1.13_frct.sigma, model1.73_frct.sigma)
ymin = min(model1.13_frct.sigma, model1.73_frct.sigma)
par(mfrow = c(1, 1))
plot.ts(model1.13_frct.sigma, main =" ", ylim=c(ymin,ymax),
col = "black", lwd=2, ylab="sigma(t+h|t)", xlab="h")
lines(model1.13_frct.sigma, col = "black", lwd = 2)
lines(model1.73_frct.sigma, col = "blue", lwd = 2)
legend(x = "topleft", legend = c("ARMA(0,0)-GARCH(1,1)~GED(0, 1, 0.941)", "ARMA(2,2)-GARCH(1,1)~GED(0, 1,
       col = c("black", "blue"), lwd = 2, lty = "solid")

#---------------------------------------------------------------------------#

# Plot of forecasted time series.
par(mfrow = c(1, 2))
plot(model1.13_frct, which = 1)
plot(model1.73_frct, which = 1)

#---------------------------------------------------------------------------#

# Plot of forecasted volatility.
par(mfrow = c(1, 2))
plot(model1.13_frct, which = 4)
plot(model1.73_frct, which = 4)

#---------------------------------------------------------------------------#
```

```
# Evaluate rolling forecasts.
# Re-fit models leaving 10 out-of-sample observations for forecast evaluation statistics.
model1.13_fit = ugarchfit(spec = btc13, data = rBTC, out.sample = 10)
model1.73_fit = ugarchfit(spec = btc73, data = rBTC, out.sample = 10)

#----------------------------------------------------------------------------#

# Compute 10 10-step ahead rolling forecasts.
model1.13_fcst = ugarchforecast(model1.13_fit, n.ahead = 10, n.roll = 10)
model1.73_fcst = ugarchforecast(model1.73_fit, n.ahead = 10, n.roll = 10)

#----------------------------------------------------------------------------#

# Compare persistence and unconditional variance.
c.mat = matrix(0, 2, 2)
colnames(c.mat) = c("Persistence", "E[sig(t)]")
rownames(c.mat) = c("ARMA(0,0)-GARCH(1,1)", "ARMA(2,2)-GARCH(1,1)")
c.mat[1,1] = persistence(model1.13_fit)
c.mat[1,2] = persistence(model1.73_fit)
c.mat[2,1] = sqrt(uncvariance(model1.13_fit))
c.mat[2,2] = sqrt(uncvariance(model1.73_fit))
c.mat

c.mat = matrix(0, 2, 2)
colnames(c.mat) = c("Persistence", "E[sig(t)]")
rownames(c.mat) = c("ARMA(0,0)-GARCH(1,1)", "ARMA(2,2)-GARCH(1,1)")
c.mat[1,1] = persistence(model1.13)
c.mat[1,2] = persistence(model1.73)
c.mat[2,1] = sqrt(uncvariance(model1.13))
c.mat[2,2] = sqrt(uncvariance(model1.73))
c.mat

#----------------------------------------------------------------------------#

# Plot the 10 10-step ahead rolling forecasts of competing models.
# Rolling forecast plots.
par(mfrow = c(1, 1))
```

```
plot(model1.13_fcst, which = 2)
plot(model1.73_fcst, which = 2)

# Rolling sigma forecasts of competing models.
plot(model1.13_fcst, which = 4)
plot(model1.73_fcst, which = 4)

###################################################
### code chunk number 9:
###################################################
# We want to look at the distribution of the residuals.
# If they are nonnormal then we have to look for other tests,
# for the regression analysis.

# Extract the residuals from the fitted model - ARMA(0,0)-GARCH(1,1)~GED.
model1.13_res = residuals(model1.13)

# Plot the residuals
plot(model1.13_res, ylab = "Residuals",  xlab = "Time", lwd = 1)
abline(h = 0, lty = 2, lwd = 2, col = "red")


#summary of residuals of fitted model
summary(model1.13_res)


# Note:
# From the residual plot we can see that the mean of the residuals is almost zero,
# but slightly negative. However the variance is not constant throughout, with
# particularly larger variation around the end of 2017.

# Testing for stationarity.
# Autocorrelation and partial autocorrelation functions.
acf(model1.13_res)

# or

plot(model1.13, which = 10)
```

```
# Note:
# The values of the ACF almost fall inside the dashed lines with slight deviations,
# at lag = 19, 20.
# It is expected that 5% of lags fall outside the two dashed lines under the null,
# hypothesis
# The figures show no significant evidence that the residuals are autocorrelated.

# For furher analysis, we plot the absolute value of the residuals.
model1.13_res_abs = abs(model1.13_res)
acf(model1.13_res_abs)

# Note:
# We find that they are obviously autocorrelated, we therefore reject the null,
# hypothesis of independence.

#----------------------------------------------------------------------------#

# Testing for the underlying distribution of the residuals, and testing for
# normality.

# 2-sided t-test
t.test(model1.13_res)

# Skewness test under the hypothesis of normality.
skewness.norm.test(model1.13_res)

# Kurtosis test under the hypothesis of normality.
kurtosis.norm.test(model1.13_res)

# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(model1.13_res)

# The histogram of the residuals will indicate the distribution.
hist(model1.13_res, nclass = 100, probability = TRUE, xlab = expression('Log returns'), ylim = c(0,30))

# Alternative plot
plot(model1.13, which = 8)
```

```
# Note:
# We see that they are approximately normal from the plot but there is still
# evidence of heavy tails.

# Q-Q plot of residuals of the fitted model
qqnorm(model1.13_res, pch = 19)
qqline(model1.13_res, col = "red", lwd = 2)

# Note:
# Shows that the distribution of the residuals have extremely heavy tails,
# so not normal.

############################################################################
############################################################################

# Extract the residuals from the fitted model - ARMA(2,2)-GARCH(1,1)~GED.
model1.73_res = residuals(model1.73)

# Plot the residuals
plot(model1.73_res, ylab = "Residuals",  xlab = "Time", lwd = 1)
abline(h = 0, lty = 2, col = "red")

#summary of residuals of fitted model
summary(model1.73_res)

# Note:
# From the residual plot we can see that the mean of the residuals is almost zero,
# but slightly negative. However the variance is not constant throughout, with
# particularly larger variation around the end of 2017.

# Testing for stationarity.
# Autocorrelation and partial autocorrelation functions.
acf(model1.73_res)

# or

plot(model1.73, which = 10)
```

```
# Note:
# The values of the ACF almost fall inside the dashed lines with slight deviations,
# at lag = 1, 19, 20.
# It is expected that 5% of lags fall outside the two dashed lines under the null,
# hypothesis.
# The figures show no significant evidence that the residuals are autocorrelated.
# This is in line with what is expected of the error term. Although the assumption,
# of the squared errors is serial autocorrelation. By analysisng the sample ACF and
# sample PACF functions of the squared errors, the assumption that the squared errors
# are correlated is confirmed.

# For furher analysis, we plot the absolute value of the residuals.
model1.73_res_abs = abs(model1.73_res)
acf(model1.73_res_abs)

# Note:
# We find that they are obviously autocorrelated, we therefore reject the null,
# hypothesis of independence.

#----------------------------------------------------------------------------#

# Testing for the underlying distribution of the residuals, and testing for
# normality.

# 2-sided t-test
t.test(model1.73_res)

# Skewness test under the hypothesis of normality.
skewness.norm.test(model1.73_res)

# Kurtosis test under the hypothesis of normality.
kurtosis.norm.test(model1.73_res)

# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(model1.73_res)
```

```
# The histogram of the residuals will indicate the distribution.
hist(model1.73_res, nclass = 100, probability = TRUE, xlab = expression('Log returns'), ylim = c(0,30))

# Alternative plot
plot(model1.73, which = 8)

# Note:
# We see that they are approximately normal from the plot but there is still
# evidence of heavy tails.

# Q-Q plot of residuals of the fitted model.
qqnorm(model1.73_res, pch = 19)
qqline(model1.73_res, col = "red", lwd = 2)

# Note:
# Shows that the distribution of the residuals have extremely heavy tails,
# so not normal.

#************************************************#
#---------------------- End ----------------------#
#************************************************#
```

Appendix C.

```
#************************************************#
# This is the relevant R code for the simulation  #
# study of our model in Chapter 5.          #
#************************************************#

###################################################
### code chunk number 1:
###################################################
# Set working directory.
setwd("C:/Users/zola/Documents/Zola/Research/Code")

###################################################
### code chunk number 2:
```

```
###################################################
# Install and load relevant packages for our simulations.

install.packages("MASS")
install.packages("fBasics")
install.packages("normtest")
install.packages("tseries")
install.packages("xts")
install.packages("maxLik")
install.packages("caret")
install.packages("quantmod")
install.packages("rugarch")
install.packages("fGarch")

# Load the packages.
library(MASS)
library(fBasics)
library(normtest)
library(tseries)
library(xts)
library(maxLik)
library(caret)
library(quantmod)
library(rugarch)
library(fGarch)

###################################################
### code chunk number 3:
###################################################
# Consider a regression model with errors modelled by a ARMA(1,1)-GARCH(1,1)
# process.
# We will give the results of on simulation run.

# Sample size.
n = 200

# Setting the seed value so as to produce pseudorandom results
# (i.e. replicatable results).
```

```
seed_val = 455
set.seed(seed_val)
print(seed_val)

#---------------------------------#
# Set mean parameters:
#---------------------------------#

# The regression coeffiecients.
beta_1 = 0.9
beta_2 = 0.01

# Create a vector mu.
# mu is vector of means for two variables.
mu_x = c(0,0)

# Create a matrix sigma_x that is a variance-covariance matrix of two variables.
# This matrix is a positive definite symmetric matrix.
sigma_x = matrix(c(1,0.5,0.5,1), 2,2) #2x2 matrix.

# Produces n observations of 2 normally distributed variables with mu and sigma,
# as the normal distribution parameters.
X_matrix = mvrnorm(n, mu_x, sigma_x)

# Create vectors to store all the iterations of Y_t, epsilon_t
y_t_vec = matrix(data=NA, nrow=n, ncol=1)
epsilon_t_vec = matrix(data=NA, nrow=n, ncol=1)
sigma_t_2_vec = matrix(data=NA, nrow=n, ncol=1)

#---------------------------------#
# Set variance parameters:
#---------------------------------#

# The underlying distribution of the model.
# eta_t  is i.i.d. standard normally distributed (i.e. eta_t ~ N(0,1) for all t).
mean_eta_t  = 0
var_eta_t  = 1
```

```
# The ARMA(1,1) parameters of the error model.
c = 1
phi_1 = 0.4
theta_1 = 0.55

# The GARCH parameters of the error model.
omega = 0.1
alpha_1 = 0.2
gamma_1 = 0.7
sigma_t1_2 = 1

# Setting the initial values neccessary for the process.
eta_t_1 = rnorm(1 ,mean_eta_t, var_eta_t)

# Note:
# Since s = max(p,q,m,n) = max(1,1,1,1) = 1, which means that there only a
# need for s=1 intial values.
# Therefore epsilon_t_1_squared = sigma_t1_2 = unconditional_variance.
epsilon_t_1 = 1

sigma_t1_2 = 1

sigma_t_2 = 1

#---------------------------------#
# One simulation run:
#---------------------------------#
# Just one simulation of data.
# A times series of n observations was generated from a ARMA(1,1)-GARCH(1,1).

# Setting the seed value so as to produce pseudorandom results
# (i.e. replicatable results).
seed_val = 455
set.seed(seed_val)
print(seed_val)

# Create an empty matrices.
matrix_data = matrix(,nrow = n,)
```

```
estimate_beta1 = matrix(data = NA, nrow = 1, ncol = 1)

estimate_beta2 = matrix(data = NA, nrow = 1, ncol = 1)


for (t in 1:n){

sigma_t = sqrt(sigma_t_2)


eta_t = rnorm(1, mean_eta_t, var_eta_t)


epsilon_t = c + ((phi_1)*(epsilon_t_1)) + ((theta_1)*(eta_t_1)) + ((sigma_t)*(eta_t))


sigma_t_2 = omega + ((alpha_1)*(epsilon_t_1**2)) + ((gamma_1)*(sigma_t1_2))


if (t > 0){

x_1 = X_matrix[t,1]

x_2 = X_matrix[t,2]

y_t = ((beta_1)*(x_1)) + ((beta_2)*(x_2)) + epsilon_t

epsilon_t_1 = epsilon_t

eta_t_1 = eta_t

sigma_t1_2 = sigma_t_2

#sigma_t = sqrt(sigma_t_2)

}

y_t_vec[[t]] = y_t

epsilon_t_vec[[t]] = epsilon_t

sigma_t_2_vec[[t]] = sigma_t_2

print(paste("t:", t, "Y_T", y_t, "epsilon is :", epsilon_t, "Var_epsilont is :", sigma_t_2))

}


matrix_data = matrix(data =cbind(y_t_vec,X_matrix[,1],X_matrix[,2],epsilon_t_vec, sigma_t_2_vec),

 nrow = n, ncol = 5)

print(head(matrix_data))


# Create weekly indicators from 03 April 2016.

week = seq(as.Date("2016-04-03"), length = length(y_t_vec), by = "weeks")


# Create a data frame with all the weekly data

all_data = data.frame(matrix_data)


# The assignment of the weekly index date to the new data frame and stored in the DataFrame.
```

```
simulation_results = xts(all_data, week)

# Print the data to view
head(simulation_results)

# Descriptive statistics of the new data frame
basicStats(simulation_results, ci = 0.95)

# Plot Y_t, epsilon_t, sigma_t_2.
par(mfrow = c(3,1))
plot(simulation_results[,1], type = "l", ylab = "Y_t", xlab = "Date", main = " ")
plot(simulation_results[,4], type = "l", ylab = "epsilon_t", xlab = "Date", main = " ")
plot(simulation_results[,5], type = "l", ylab = "sigma_t_2", xlab = "Date", main = " ")

# Likelihood function.
logLikFun = function(param){
T = length(simulation_results[,1])
beta_1 = param[1]
beta_2 = param[2]

mu = mean(simulation_results[,4])

epsilon_tt = simulation_results[,1] - (((beta_1)*(simulation_results[,2])) +
((beta_2)*(simulation_results[,3])))
sigma_t_22 = simulation_results[,5]

ll = (-0.5*T*log(2*pi)) - ((0.5)*sum(log(sigma_t_2))) -
 ((0.5)*(sum(((epsilon_tt-mu)**2)/(sigma_t_22))))

}
# Maximum log-likelihood estimation (MLE).
mle_trail = maxLik(logLik = logLikFun, start = c(beta_1 = 0, beta_2 = 0))
summary(mle_trail)

# Storing the estimated values.
estimate_beta1 = coef(mle_trail)[1]
estimate_beta2 = coef(mle_trail)[2]
matrix_data_beta = matrix(data = cbind(estimate_beta1, estimate_beta2))
```

```
print(head(matrix_data_beta))

#--------------------------------#
# One simulation run log returns:
#--------------------------------#
# Defining the log return time series data.

# Time series of simulated log retunrns.
rY_t = log(simulation_results[,1]/lag(simulation_results[,1], -1))
summary(rY_t)

# Removal of the last missing value.
rY_t = na.omit(rY_t)
summary(rY_t)

# Strip off dates and create a numeric object.
log_returns_Y_t = coredata(rY_t)

# Descriptive statistics of simulated log returns.
basicStats(rY_t, ci = 0.95)

# Plot y_t, rY_t.
par(mfrow = c(2,1))
plot(simulation_results[,1], type = "l", ylab = "Y_t", xlab = "Date", main = " ")
title(main = Y[t]~" time series", xlab = "Date", ylab = expression(Y[t]))
plot(rY_t, type = "l", ylab = "rY_t", xlab = "Date", main = " ")
title(main = "Log returns of "~Y[t]~" time series", xlab = "Date", ylab = expression(log(Y[t])))

#-------------------------------------------#
# Testing the GARCH effect on the residuals:
#-------------------------------------------#
# Fitting a linear model to the simulated series. No intercept.
model_ols = lm(formula = X1 ~ X2 + X3 - 1, data = simulation_results)
summary(model_ols)

# Extract the residuals of the OLS model.
model_residuals = model_ols$residuals
```

```
# Descriptive statistics of the OLS residuals.
basicStats(model_residuals, ci = 0.95)

# Plot residuals.
par(mfrow = c(1,1))
plot(model_residuals, col = 1, xlab = "Week", ylab = "Residuals", main = "OLS Residuals Plot")
abline(h = 0,lty = 2, col = 2)

# Plot the model residuals.
plot.ts(model_residuals, ylab = expression(epsilon[t]),  xlab = "Time Index", col = 1,
lwd = 1, main = "BTC_price ~ BTC_interest + SP500  linear model residuals")
abline(h = 0,lty = 2, col = 2)



# Testing for stationarity.
# Autocorrelation and partial autocorrelation functions.
par(mfrow = c(2,1))
acf(model_residuals)
pacf(model_residuals)

# Augmented Dickey-Fuller Test for non-staionarity of the OLS residuals.
adf.test(model_residuals)

#------------------------------------------------------------------------------#

# Testing for the underlying distribution of the OLS residuals, and testing for
# normality.

# 2-sided t-test
t.test(model_residuals)

# Skewness test under the hypothesis of normality of the OLS residuals.
skewness.norm.test(model_residuals)

# Kurtosis test under the hypothesis of normality of the OLS residuals.
kurtosis.norm.test(model_residuals)
```

```
# Jarque-Bera (JB) test under the hypothesis of normality of the OLS residuals.
jb.norm.test(model_residuals)

#-----------------------------------------------------------------------------#


#Q-Q plot of the OLS residuals.
par(mfrow = c(1,1))
qqnorm(model_residuals, pch = 19)
qqline(model_residuals, col = "red", lwd = 2)

#-----------------------------------------------------------------------------#


# Plot a histogram of the OLS residuals.
par(mfrow = c(1,1))
xfit1 = seq(min(model_residuals), max(model_residuals), length = length(model_residuals))
yfit1 = dnorm(xfit1, mean = mean(model_residuals), sd = sd(model_residuals))

# Histogram of OLS residuals.
hist(model_residuals, nclass = 20, probability = TRUE, xlab = "Residuals",
 main = "Histogram of residuals")
#lines(density(model_residuals), lwd = 2, col = "red")

#fitting a normal distribution curve with the actual true values of the data.
lines(xfit1, yfit1, lwd = 2, col = "blue")

#-----------------------------------------------------------------------------#
#Modelling the residuals with an intercept  ARMA(0,0)-GARCH(1,1) errors.
model_spec1 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(0,0), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted1 = ugarchfit(spec = model_spec1 , data = model_residuals)
model_fitted1

#####

#Modelling the residuals with an intercept  ARMA(1,0)-GARCH(1,1) errors.
model_spec2 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,0), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted2 = ugarchfit(spec = model_spec2 , data = model_residuals)
model_fitted2

#####

#Modelling the residuals with an intercept  ARMA(0,1)-GARCH(1,1) errors.
model_spec3 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(0,1), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted3 = ugarchfit(spec = model_spec3 , data = model_residuals)
model_fitted3

#####

#Modelling the residuals with an intercept  ARMA(1,1)-GARCH(1,1) errors.
model_spec4 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,1), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted4 = ugarchfit(spec = model_spec4, data = model_residuals)
model_fitted4

#####

#Modelling the residuals with an intercept  ARMA(2,1)-GARCH(1,1) errors.
model_spec5 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,1), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted5 = ugarchfit(spec = model_spec5, data = model_residuals)
model_fitted5

#####
```

```
#Modelling the residuals with an intercept  ARMA(1,2)-GARCH(1,1) errors.
model_spec6 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,2), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted6 = ugarchfit(spec = model_spec6, data = model_residuals)
model_fitted6

#####

#Modelling the residuals with an intercept  ARMA(2,2)-GARCH(1,1) errors.
model_spec7 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,2), include.mean = TRUE), distribution.model = "norm")

#The model fitting
model_fitted7 = ugarchfit(spec = model_spec7, data = model_residuals)
model_fitted7

#--------------------------------------------#
# Squared residuals analysis:
#--------------------------------------------#
model_residuals_vec = matrix(data = cbind(model_residuals), ncol = 1)
model_residuals_vec_sq = model_residuals_vec**2

# Testing for stationarity.
# Autocorrelation and partial autocorrelation functions.
par(mfrow = c(2,1))
acf(model_residuals_vec_sq)
pacf(model_residuals_vec_sq)

###################################################
### code chunk number 4:
###################################################
# Consider a regression model with errors modelled by a ARMA(1,1)-GARCH(1,1) process.
# We will perform N simulations and determine the best parameter estimates of the
# regression coefficients by way of linear shrinkage.
# The results will be given below.
# Number of simulations.
N_sim = 10000

# Sample size.
n = 200

# Setting the seed value so as to produce pseudorandom results (i.e. replicatable results).
seed_val = 456
set.seed(seed_val)
print(seed_val)

#---------------------------------#
# Set mean parameters:
#---------------------------------#

# The regression coeffiecients.
beta_1 = 0.9
beta_2 = 0.01

# Create vectors to store all the iterations of Y_t, epsilon_t
y_t_vec = matrix(data=NA, nrow=n, ncol=1)
epsilon_t_vec = matrix(data=NA, nrow=n, ncol=1)
sigma_t_2_vec = matrix(data=NA, nrow=n, ncol=1)

#---------------------------------#
# Set variance parameters:
#---------------------------------#

# The underlying distribution of the model.
# eta_t  is i.i.d. standard normally distributed (i.e. eta_t ~ N(0,1) for all t).
mean_eta_t  = 0
var_eta_t  = 1

# The ARMA(1,1) parameters of the error model.
c = 1
phi_1 = 0.4
theta_1 = 0.55
```

```
# The GARCH parameters of the error model.
omega = 0.1
alpha_1 = 0.2
gamma_1 = 0.7
sigma_t1_2 = 1

# Setting the initial values neccessary for the process.
eta_t_1 = rnorm(1 ,mean_eta_t, var_eta_t)

# Note:
# Since s = max(p,q,m,n) = max(1,1,1,1) = 1, which means that there only a need for s=1,
# intial values.
# Therefore epsilon_t_1_squared = sigma_t1_2 = unconditional_variance.
epsilon_t_1 = 1

sigma_t1_2 = 1

sigma_t_2 = 1

#---------------------------------#
# N simulation run:
#---------------------------------#
# We will simulate N time series.
# Each times series has n observations was generated from a ARMA(1,1)-GARCH(1,1).

# Intitialize vectors that will store the beta estimates.
# We have the true beta estimates.
vec_data_beta1_true = matrix(data = beta_1, nrow = N_sim, ncol = 1)
vec_data_beta2_true = matrix(data = beta_2, nrow = N_sim, ncol = 1)

# We have the unrestricted beta estimates.
vec_data_beta1_ur = matrix(data = NA, nrow = 2, ncol = 1)
vec_data_beta2_ur = matrix(data = NA, nrow = 2, ncol = 1)

# We have the restricted beta estimates - Option 1.
vec_data_beta1_r_option1 = matrix(data = 1, nrow = N_sim, ncol = 1)
vec_data_beta2_r_option1 = matrix(data = 0, nrow = N_sim, ncol = 1)
```

```
# We have the restricted beta estimates - Option 2.
vec_data_beta1_r_option2 = matrix(data = 0.001, nrow = N_sim, ncol = 1)
vec_data_beta2_r_option2 = matrix(data = 2, nrow = N_sim, ncol = 1)

# We have the restricted beta estimates - Option 3.
vec_data_beta1_r_option3 = matrix(data = 0.91, nrow = N_sim, ncol = 1)
vec_data_beta2_r_option3 = matrix(data = 0.02, nrow = N_sim, ncol = 1)

# The simulations.
for (i in 1:N_sim){
# Set the seed value
seed_val = 455 + i
set.seed(seed_val)
print(seed_val)

# Create a vector mu.
# mu is vector of means for two variables.
mu_x = c(0,0)

# Create a matrix sigma_x that is a variance-covariance matrix of two variables.
# This matrix is a positive definite symmetric matrix.
sigma_x = matrix(c(1,0.5,0.5,1), 2,2) #2x2 matrix.

# Produces n observations of 2 normally distributed variables with mu and sigma,
# as the normal distribution parameters.
X_matrix = mvrnorm(n, mu_x, sigma_x)

# Create an empty matrix.
matrix_data = matrix(,nrow = n, ncol = 5)

for (t in 1:n){
sigma_t = sqrt(sigma_t_2)

eta_t = rnorm(1, mean_eta_t , var_eta_t)

epsilon_t = c + ((phi_1)*(epsilon_t_1)) + ((theta_1)*(eta_t_1)) + ((sigma_t)*(eta_t))

sigma_t_2 = omega + ((alpha_1)*(epsilon_t_1**2)) + ((gamma_1)*(sigma_t1_2))
```

```
if (t > 0){
x_1 = X_matrix[t,1]
x_2 = X_matrix[t,2]
y_t = ((beta_1)*(x_1)) + ((beta_2)*(x_2)) + epsilon_t
epsilon_t_1 = epsilon_t
eta_t_1 = eta_t
sigma_t1_2 = sigma_t_2
#sigma_t = sqrt(sigma_t_2)
}

y_t_vec[[t]] = y_t
epsilon_t_vec[[t]] = epsilon_t
sigma_t_2_vec[[t]] = sigma_t_2
print(paste("t:", t, "Y_T", y_t, "epsilon is :", epsilon_t, "Var_epsilont is :", sigma_t_2))
}
matrix_data = matrix(data =c(y_t_vec,X_matrix[,1],X_matrix[,2],epsilon_t_vec, sigma_t_2_vec),
 nrow = n, ncol = 5)
print(head(matrix_data))

#-----------------------------------------------#
# Store the matrix information in a DataFrame:
#-----------------------------------------------#
# Create weekly indicators from 03 April 2016
week = seq(as.Date("2016-04-03"), length = length(y_t_vec), by = "weeks")

# Create a data frame with all the weekly data.
all_data = data.frame(matrix_data)

# The assignment of the weekly index date to the new data frame and stored in the DataFrame.
simulation_results = xts(all_data, week)

# Print the data to view.
head(simulation_results)

# Plot y_t
par(mfrow = c(2,1))
plot(simulation_results[,1], type = "l", col = 1, autogrid = FALSE, ylab = "Price",
 xlab = "Date", main = "Y_t Simulation")

# Descriptive statistics of the new data frame.
basicStats(simulation_results, ci = 0.95)

# Log-likelihood function.
logLikFun = function(param) {
T = length(simulation_results[,1])
beta_1 = param[1]
beta_2 = param[2]

mu = mean(simulation_results[,4])

epsilon_tt = simulation_results[,1] - (((beta_1)*(simulation_results[,2])) + ((beta_2)*(simulation_results
sigma_t_22 = simulation_results[,5]

ll = (-0.5*T*log(2*pi)) - ((0.5)*sum(log(sigma_t_2))) - ((0.5)*(sum(((epsilon_tt-mu)**2)/(sigma_t_22))))

}
# Maximum log-likelihood estimation (MLE).
mle_trail = maxLik(logLik = logLikFun, start = c(beta_1 = 0, beta_2 = 0))
summary(mle_trail)
#coef(mle_trail)
vec_data_beta1_ur[[i]] = coef(mle_trail)[1]
vec_data_beta2_ur[[i]] = coef(mle_trail)[2]

}

# Matrix of unrestricted beta estimates.
matrix_data_beta_ur = matrix(data = cbind(vec_data_beta1_ur, vec_data_beta2_ur),
 nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_ur))

# Matrix of true beta estimates.
matrix_data_beta_true0 = matrix(data = cbind(vec_data_beta1_true, vec_data_beta2_true),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_true0))
```

```
# Option 1 - matrix of restricted estimators to be tested.
matrix_data_beta_r_option1 = matrix(data = cbind(vec_data_beta1_r_option1, vec_data_beta2_r_option1),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_r_option1))

# Option 2 - matrix of restricted estimators to be tested.
matrix_data_beta_r_option2 = matrix(data = cbind(vec_data_beta1_r_option2, vec_data_beta2_r_option2),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_r_option2))

# Option 3 - matrix of restricted estimators to be tested.
matrix_data_beta_r_option3 = matrix(data = cbind(vec_data_beta1_r_option3, vec_data_beta2_r_option3),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_r_option3))

#-----------------------------------------------#
# Computing the shrinkage estimates - Option 1:
#-----------------------------------------------#
# pi values grid.
pi_values = c(0, 0.01, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,
  0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1)

# (1-pi) values grid.
one_minus_pi_values = 1-pi_values

# Create an empty matrix for the beta shrinkage estimates.
matrix_data_beta_shrinkage1 = matrix(data = NA, nrow = N_sim, ncol = 2)

# Create an empty matrix for the beta shrinkage estimate MSE values.
MSE_pi_values_option1 = matrix(data = NA , nrow = length(pi_values), ncol = 3)

# Compute the MSE values.
for (i in 1:length(pi_values)){
matrix_data_beta_shrinkage1[,1] = (pi_values[[i]]*matrix_data_beta_ur[,1])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r_option1[,1])
matrix_data_beta_shrinkage1[,2] = (pi_values[[i]]*matrix_data_beta_ur[,2])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r_option1[,2])
MSE_pi_values_option1[i,1] = pi_values[[i]]
MSE_pi_values_option1[i,2] = one_minus_pi_values[[i]]
MSE_pi_values_option1[i,3] = (1/N_sim)*
 (sum((t(matrix_data_beta_shrinkage1-matrix_data_beta_true0))%*%
 (matrix_data_beta_shrinkage1-matrix_data_beta_true0)))
}

# Print the MSE values for each pi value chosen.
print(MSE_pi_values_option1)

#----------------------------------------------#
# Computing the shrinkage estimates - Option 2:
#----------------------------------------------#
# pi values grid.
pi_values = c(0, 0.01, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,
  0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1)

# (1-pi) values grid.
one_minus_pi_values = 1-pi_values

# Create an empty matrix for the beta shrinkage estimates.
matrix_data_beta_shrinkage2 = matrix(data = NA, nrow = N_sim, ncol = 2)

# Create an empty matrix for the beta shrinkage estimate MSE values.
MSE_pi_values_option2 = matrix(data = NA , nrow = length(pi_values), ncol = 3)

# Compute the MSE values.
for (i in 1:length(pi_values)){
matrix_data_beta_shrinkage2[,1] = (pi_values[[i]]*matrix_data_beta_ur[,1])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r_option2[,1])
matrix_data_beta_shrinkage2[,2] = (pi_values[[i]]*matrix_data_beta_ur[,2])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r_option2[,2])
MSE_pi_values_option2[i,1] = pi_values[[i]]
MSE_pi_values_option2[i,2] = one_minus_pi_values[[i]]
MSE_pi_values_option2[i,3] = (1/N_sim)*
 (sum((t(matrix_data_beta_shrinkage2-matrix_data_beta_true0))%*%
 (matrix_data_beta_shrinkage2-matrix_data_beta_true0)))
}
```

```
# Print the MSE values for each pi value chosen.
print(MSE_pi_values_option2)

#----------------------------------------------#
# Computing the shrinkage estimates - Option 3:
#----------------------------------------------#
# pi values grid.
pi_values = c(0, 0.01, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,
  0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1)

# (1-pi) values grid.
one_minus_pi_values = 1-pi_values

# Create an empty matrix for the beta shrinkage estimates.
matrix_data_beta_shrinkage3 = matrix(data = NA, nrow = N_sim, ncol = 2)

# Create an empty matrix for the beta shrinkage estimate MSE values.
MSE_pi_values_option3 = matrix(data = NA , nrow = length(pi_values), ncol = 3)

# Compute the MSE values.
for (i in 1:length(pi_values)){
matrix_data_beta_shrinkage3[,1] = (pi_values[[i]]*matrix_data_beta_ur[,1])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r_option3[,1])
matrix_data_beta_shrinkage3[,2] = (pi_values[[i]]*matrix_data_beta_ur[,2])+
      (one_minus_pi_values[[i]]*matrix_data_beta_r_option3[,2])
MSE_pi_values_option3[i,1] = pi_values[[i]]
MSE_pi_values_option3[i,2] = one_minus_pi_values[[i]]
MSE_pi_values_option3[i,3] = (1/N_sim)*
 (sum((t(matrix_data_beta_shrinkage3-matrix_data_beta_true0))%*%
 (matrix_data_beta_shrinkage3-matrix_data_beta_true0)))
}

# Print the MSE values for each pi value chosen.
print(MSE_pi_values_option3)

#---------------------------------------------------#
# The optimal linear shrinkage estimates - Option 1:
#---------------------------------------------------#
```

```
# Create an empty matrix for the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal1 = matrix(data=NA, nrow = N_sim, ncol = 2)

# The optimal pi value.
pi_optimal1 = 0.10

# The optimal (1-pi) value.
one_minus_pi_optimal1 = 1-pi_optimal1

# Store the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal1[,1] = (pi_optimal1*matrix_data_beta_ur[,1])+
  (one_minus_pi_optimal1*matrix_data_beta_r_option1[,1])
matrix_data_beta_shrinkage_optimal1[,2] = (pi_optimal1*matrix_data_beta_ur[,2])+
  (one_minus_pi_optimal1*matrix_data_beta_r_option1[,2])
print(head(matrix_data_beta_shrinkage_optimal1))

# Descriptive statistics of the optimal beta shrinkage estimates.
basicStats(matrix_data_beta_shrinkage_optimal1, ci = 0.95)

# Plot histograms of each of the optimal beta shrinkage estimate’s distribution.
# Histogram of Beta_1 distribution.
par(mfrow = c(1,1))
hist(matrix_data_beta_shrinkage_optimal1[,1], nclass = 20, probability = TRUE,
 xlab = expression(beta[1]), xlim = c(0.8,1.2), ylim = c(0,20), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal1[,1]), lwd = 2, col = "red")

# Histogram of Beta_2 distribution.
hist(matrix_data_beta_shrinkage_optimal1[,2], nclass = 20, probability = TRUE,
 xlab = expression(beta[2]),  xlim = c(-0.2,0.2), ylim = c(0,20), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal1[,2]), lwd = 2, col = "blue")

# Test for normality
# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(matrix_data_beta_shrinkage_optimal1)

# Q-Q plot of log returns distribution.
qqnorm(matrix_data_beta_shrinkage_optimal1, pch = 19)
qqline(matrix_data_beta_shrinkage_optimal1, col = "red", lwd = 2)
```

```
#---------------------------------------------------#
# The optimal linear shrinkage estimates - Option 2:
#---------------------------------------------------#
# Create an empty matrix for the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal2 = matrix(data=NA, nrow = N_sim, ncol = 2)

# The optimal pi value.
pi_optimal2 = 0.95

# The optimal (1-pi) value.
one_minus_pi_optimal2 = 1-pi_optimal2

# Store the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal2[,1] = (pi_optimal2*matrix_data_beta_ur[,1])+
  (one_minus_pi_optimal2*matrix_data_beta_r_option2[,1])
matrix_data_beta_shrinkage_optimal2[,2] = (pi_optimal2*matrix_data_beta_ur[,2])+
  (one_minus_pi_optimal2*matrix_data_beta_r_option2[,2])
print(head(matrix_data_beta_shrinkage_optimal2))

# Descriptive statistics of the optimal beta shrinkage estimates.
basicStats(matrix_data_beta_shrinkage_optimal2, ci = 0.95)

# Plot histograms of each of the optimal beta shrinkage estimate’s distribution.
# Histogram of Beta_1 distribution.
par(mfrow = c(1,2))
hist(matrix_data_beta_shrinkage_optimal2[,1], nclass = 50, probability = TRUE,
 xlab = expression(beta[1]), xlim = c(-0.4,2.4), ylim = c(0,2), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal2[,1]), lwd = 2, col = "red")

# Histogram of Beta_2 distribution.
hist(matrix_data_beta_shrinkage_optimal2[,2], nclass = 50, probability = TRUE,
 xlab = expression(beta[2]),  xlim = c(-1.2,1.5), ylim = c(0,2), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal2[,2]), lwd = 2, col = "blue")

# Test for normality
# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(matrix_data_beta_shrinkage_optimal2)
```

```
# Q-Q plot of log returns distribution.
qqnorm(matrix_data_beta_shrinkage_optimal2, pch = 19)
qqline(matrix_data_beta_shrinkage_optimal2, col = "red", lwd = 2)

#---------------------------------------------------#
# The optimal linear shrinkage estimates - Option 3:
#---------------------------------------------------#
# Create an empty matrix for the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal3 = matrix(data=NA, nrow = N_sim, ncol = 2)

# The optimal pi value.
pi_optimal3 = 0.01

# The optimal (1-pi) value.
one_minus_pi_optimal3 = 1-pi_optimal3

# Store the optimal beta shrinkage estimates.
matrix_data_beta_shrinkage_optimal3[,1] = (pi_optimal3*matrix_data_beta_ur[,1])+
  (one_minus_pi_optimal3*matrix_data_beta_r_option3[,1])
matrix_data_beta_shrinkage_optimal3[,2] = (pi_optimal3*matrix_data_beta_ur[,2])+
  (one_minus_pi_optimal3*matrix_data_beta_r_option3[,2])
print(head(matrix_data_beta_shrinkage_optimal3))

# Descriptive statistics of the optimal beta shrinkage estimates.
basicStats(matrix_data_beta_shrinkage_optimal3, ci = 0.95)

# Plot histograms of each of the optimal beta shrinkage estimate’s distribution.
# Histogram of Beta_1 distribution.
par(mfrow = c(1,2))
hist(matrix_data_beta_shrinkage_optimal3[,1], nclass = 50, probability = TRUE,
 xlab = expression(beta[1]), xlim = c(0.89,0.93), ylim = c(0,200), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal3[,1]), lwd = 2, col = "red")

# Histogram of Beta_2 distribution.
hist(matrix_data_beta_shrinkage_optimal3[,2], nclass = 50, probability = TRUE,
xlab = expression(beta[2]),  xlim = c(0, 0.04), ylim = c(0,200), main = ’’)
lines(density(matrix_data_beta_shrinkage_optimal3[,2]), lwd = 2, col = "blue")
```

```
# Test for normality
# Jarque-Bera (JB) test under the hypothesis of normality.
jb.norm.test(matrix_data_beta_shrinkage_optimal3)

# Q-Q plot of log returns distribution.
qqnorm(matrix_data_beta_shrinkage_optimal3, pch = 19)
qqline(matrix_data_beta_shrinkage_optimal3, col = "red", lwd = 2)

###################################################
### code chunk number 5:
###################################################
# The preliminary test.
# Consider the N simulations that were run and the regression coefficients were estimated for
# the unrestricted model.
# We will then determine the 100(1-alpha)% confidence interval of each regression coefficient
# estimate for different levels of significance and  then determine whether the restricted or
# unrestricted model will be chosen for each case.
# If zero is found within the confidence interval of Beta_2, then we do not reject the null
# hypothesis (i.e. restricted model chosen over the unrestricted model).
# pi values grid.

# Confidence intervals of for different alpha values (i.e. level of significance).
# alpha values grid values.
alpha_values = c(0.01, 0.025, 0.1, 0.25, 0.5)

# (1-alpha_values) values grid.
one_minus_alpha_values = 1-alpha_values

#--------------------------------------------------#
# The optimal linear shrinkage estimates:
#--------------------------------------------------#

# Create an empty matrix for the confidence interval limits of the unrestricted beta
# estimate values.
CI_ll_ul_limits_beta_1_ur = matrix(data = NA , nrow = length(alpha_values), ncol = 3)
CI_ll_ul_limits_beta_2_ur = matrix(data = NA , nrow = length(alpha_values), ncol = 3)
```

```
# Compute the confidence intervals.
for (i in 1:length(alpha_values)){
# The mean values of the unrestricted parameter estimates.
mean_beta_1_ur = basicStats(matrix_data_beta_ur, ci = one_minus_alpha_values[[i]])[7,1]
mean_beta_2_ur = basicStats(matrix_data_beta_ur, ci = one_minus_alpha_values[[i]])[7,2]
print(mean_beta_2_ur)
# The standard deviation values of the unrestricted parameter estimates.
sd_beta_1_ur = basicStats(matrix_data_beta_ur, ci = one_minus_alpha_values[[i]])[14,1]
sd_beta_2_ur = basicStats(matrix_data_beta_ur, ci = one_minus_alpha_values[[i]])[14,2]
print(sd_beta_2_ur)

# The 100(1-alpha/2)quantile values of the unrestricted parameter estimates.
quantile_value = 1 - (alpha_values[[i]]/2)
z_alpha_div_by_2 = qnorm(quantile_value)

# The margins of error values of the unrestricted parameter estimates.
moe_beta_1_ur = z_alpha_div_by_2*(sd_beta_1_ur/(sqrt(N_sim)))
moe_beta_2_ur = z_alpha_div_by_2*(sd_beta_2_ur/(sqrt(N_sim)))
print(moe_beta_2_ur)

# Lower and upper confidence limits for Beta_1
CI_ll_ul_limits_beta_1_ur[i,1] = one_minus_alpha_values[[i]]*100
CI_ll_ul_limits_beta_1_ur[i,2] = mean_beta_1_ur - moe_beta_1_ur
CI_ll_ul_limits_beta_1_ur[i,3] = mean_beta_1_ur + moe_beta_1_ur

# Lower and upper confidence limits for Beta_2
CI_ll_ul_limits_beta_2_ur[i,1] = one_minus_alpha_values[[i]]*100
CI_ll_ul_limits_beta_2_ur[i,2] = mean_beta_2_ur - moe_beta_2_ur
CI_ll_ul_limits_beta_2_ur[i,3] = mean_beta_2_ur + moe_beta_2_ur

}

# Print the confidence interval limits of the unrestricted beta estimate values.
print(CI_ll_ul_limits_beta_1_ur)
print(CI_ll_ul_limits_beta_2_ur)

#************************************************#
#---------------------- End ----------------------#
```

```
#*************************************************#
```

Appendix D.

```
#*************************************************#
# This is the relevant R code for the real data   #
# application study of our model in Chapter 5.   #
#*************************************************#

###################################################
### code chunk number 1:
###################################################
# Set working directory.
setwd("C:/Users/zola/Documents/Zola/Research/Code")

###################################################
### code chunk number 2:
###################################################
# Install and load relevant packages for our simulations.

install.packages("coinmarketcapr")
install.packages("crypto")
install.packages("MASS")
install.packages("tseries")
install.packages("ggplot2")
install.packages("fBasics")
install.packages("normtest")
install.packages("xts")
install.packages("reshape2")
install.packages("maxLik")
install.packages("caret")
install.packages("quantmod")
install.packages("rugarch")
install.packages("fGarch")
install.packages("stats4")
```

```
# Load the packages.
library(coinmarketcapr)
library(crypto)
library(MASS)
library(tseries)
library(ggplot2)
library(fBasics)
library(normtest)
library(xts)
library(reshape2)
library(maxLik)
library(caret)
library(quantmod)
library(rugarch)
library(fGarch)
library(stats4)

###################################################
### code chunk number 3:
###################################################
# We want to test whether the prior information of Google searches of Bitcoin
# and the S&P 500 index have any impact on the price of Bitcoin provided that
# the error model is ARMA-GARCH.

################################
# Reading all the relevant data:
################################

#---------------------------------#
# Bitcoin price data:
#---------------------------------#
# Read the data into R session obatined from Investing.com.
# Weekly price data of Bitcoin - BTC/USD.
BTCweek = read.csv("bitcoinweek.csv", header = T)

#---------------------------------#
# Google searches data:
```

```
#--------------------------------#
# Weekly Google searches of bitcoin - on a 0-100 scale.
# Sourced from Google Trends.
BTCsearches = read.csv("bitcointrend.csv", header = T)

#--------------------------------#
# S&P 500 index data:
#--------------------------------#
# Weekly price data of the S&P 500 index in USD.
# Obtained using "quantmod" package to get daily data and then converted to weekly data.
# This data was sourced from Yahoo finance.

# Loading the "quantmod" package which is required to obtain the the S&P 500 data
# sourced from Yahoo Finance.
require("quantmod")

SP500daily = new.env()

getSymbols("^GSPC", env = SP500daily, src = "yahoo", from = as.Date("2016-03-31"),
to = as.Date("2018-03-25"), auto.assign = TRUE)
GSPC = SP500daily$GSPC
head(GSPC)

#--------------------------------#
# All the data:
#--------------------------------#
# Dates start from the the next Monday after 31/03/2016 up until 25/03/2018.
nextmon = function(x) 7 * ceiling(as.numeric(x - 5 + 4)/7) + as.Date(7 - 4)
SP500week = xts(aggregate(GSPC, nextmon, tail, 1))

# Print the data to view.
head(BTCweek)
head(BTCsearches)
head(SP500week)

# Print summary of all data sets.
summary(BTCweek)
summary(BTCsearches)
summary(SP500week)

# Basic statistics of all variables.
basicStats(BTCweek$Price, ci = 0.95)
basicStats(BTCsearches$Searches, ci = 0.95)
basicStats(Cl(SP500week), ci = 0.95)

# Assignment of weekly index date to all data.
# Define the weekly index dates.
week = seq(as.Date("2016-04-03"), length = length(BTCweek$Price), by = "weeks")

# Create a data frame with all the weekly data.
all_data_data = data.frame(BTC_price = BTCweek$Price, BTC_interest = BTCsearches$Searches, SP500_Price = Cl

# The assignment of the weekly index date to the new data frame.
BTC_SP_week = xts(all_data_data, week)

# Print the data to view.
head(BTC_SP_week)

# Descriptive statistics of the new data frame.
basicStats(BTC_SP_week, ci = 0.95)

# plot all_data.
par(mfrow = c(3,1))
plot(BTC_SP_week[,1], type = "l", col = 1, autogrid = FALSE, ylab = "Price", xlab = "Date",
 main = "Bitcoin")
plot(BTC_SP_week[,2], type = "l", col = 2, autogrid = FALSE, ylab = "Price", xlab = "Date",
 main = "Google Searches of Bitcoin")
plot(BTC_SP_week[,3], type = "l", col = 3, autogrid = FALSE, xy.labels = c("Date", "Price, $"),
 main = "S&P 500 Index")

###################################################
### code chunk number 4:
###################################################
# In order to estimate the appropriate model for the errors, the residuals need to
# be isolated. This is done be estimating the mean model using Ordinary Least Squares
# (i.e. OLS).
# We can then easily obtain the residuals of the OLS model and fit an ARMA-GARCH
# process to the residuals (i.e. determine the error model).

# We fit a linear regression model to the real data. No intercept.
# The Bitcoin price being the dependent variable and the 2 independent variables being
# the Google searches of Bitcoin and the S&P 500 index.
model_ols = lm(formula = BTC_price ~ BTC_interest + GSPC.Close - 1, data = BTC_SP_week)
summary(model_ols)

# Extract the residuals of the OLS model.
model_residuals = model_ols$residuals

# Descriptive statistics of the OLS residuals.
basicStats(model_residuals, ci = 0.95)

# Plot the model residuals.
par(mfrow = c(1,1))
model_residuals_df = data.frame(model_residuals, week)
model_residuals_df$week <- as.Date(week)
require(scales)
ggplot(model_residuals_df, aes(x = week, y = model_residuals)) +
geom_line(color = "#0B0B0B", size = 2) + xlab("Week, t") +ylab("Residuals, "~epsilon[t])+
geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1)+
scale_x_date(labels = date_format("%b %d %Y"), breaks = date_breaks("2 month"))+
scale_y_continuous(limits = c(-7000, 7000), breaks = c(-6000, -4000, -2000, 0, 2000, 4000, 6000))+
ggsave("time_plot.png", height = 4, width = 5)

# Plot the model residuals.
plot(model_residuals, xlab = "Week", ylab = expression(epsilon[t]), col = 1, lwd = 1,
 main = "OLS Residuals Plot")
abline(h = 0,lty = 2, col = 2)

# Testing for stationarity.
# Autocorrelation and partial autocorrelation functions.
par(mfrow = c(2,1))
acf(model_residuals)
pacf(model_residuals)
```

```
# Augmented Dickey-Fuller Test for non-staionarity of the intercept model.
adf.test(model_residuals)

#------------------------------------------------------------------------------#

# Testing for the underlying distribution of the OLS residuals, and testing for
# normality.

# 2-sided t-test
t.test(model_residuals)

# Skewness test under the hypothesis of normality of the OLS residuals.
skewness.norm.test(model_residuals)

# Kurtosis test under the hypothesis of normality of the OLS residuals.
kurtosis.norm.test(model_residuals)

# Jarque-Bera (JB) test under the hypothesis of normality of the OLS residuals.
jb.norm.test(model_residuals)

#------------------------------------------------------------------------------#

# Plot a histogram of the OLS residuals.
par(mfrow = c(1,1))
xfit1 = seq(min(model_residuals), max(model_residuals), length = length(model_residuals))
yfit1 = dnorm(xfit1, mean = mean(model_residuals), sd = sd(model_residuals))

# Histogram of OLS residuals.
hist(model_residuals, nclass = 20, probability = TRUE, xlab = "Residuals",
 main = "Histogram of residuals")
#lines(density(model_residuals), lwd = 2, col = "red")

#fitting a normal distribution curve with the actual true values of the data.
lines(xfit1, yfit1, lwd = 2, col = "blue")

#------------------------------------------------------------------------------#

#Q-Q plot of the OLS residuals.
```

```
qqnorm(model_residuals, pch = 19)
qqline(model_residuals, col = "red", lwd = 2)

#-------------------------------------------------------------#
# Removal of outliers:
#-------------------------------------------------------------#
# Create a matrix that will store the lower and upper bounds for the outliers.
model_residuals = matrix(data = c(model_residuals), ncol = 1)
model_residuals_outliers_ll = matrix(data = NA, ncol = 1, nrow = 1)
model_residuals_outliers_ul = matrix(data = NA, ncol = 1, nrow = 1)

# Determine the outliers based on the formula.
for (i in 1:ncol(model_residuals)){
# lower and upper quartiles.
q1 = basicStats(model_residuals, ci = 0.95)[5,i]
q3 = basicStats(model_residuals, ci = 0.95)[6,i]

# Interquartile range.
iqr_range = q3-q1

# lower outlier limit
model_residuals_outliers_ll[i,1] = q1 - (1.5*(iqr_range))

# upper outlier limit
model_residuals_outliers_ul[i,1] = q3 + (1.5*(iqr_range))
}

print(model_residuals_outliers_ll)
print(model_residuals_outliers_ul)

outlier_analysis_matrix = matrix(data = NA, ncol = 1, nrow = nrow(model_residuals))

# If any of the elements are greater than their respective outlier upper bounds then
# the element is given a value of 1 otherwise is given a zero.
for (i in 1:nrow(model_residuals)){
if ((model_residuals[i] > model_residuals_outliers_ll[1])&&
(model_residuals[i] < model_residuals_outliers_ul[1])){
outlier_analysis_matrix[i,1] = 0
} else {
outlier_analysis_matrix[i,1] = 1
}
}
print(outlier_analysis_matrix)

# We then determine the index values of the rows that have at least one outlier in
# the response or independent variable.
outlier_index_list = matrix(data = NA, ncol = 1)

for (i in 1:nrow(outlier_analysis_matrix)){
if (outlier_analysis_matrix[i] > 0){
outlier_index_list[i] = i
}
}
outlier_index_list2 = na.omit(outlier_index_list)
print(outlier_index_list2)

outlier_index_vec = matrix(data = outlier_index_list2, ncol = 1)
print(outlier_index_vec)

# New OLS residuals data frame.
model_residuals_df = data.frame(model_residuals)
model_residuals_new_df = matrix(data = cbind(model_residuals_df[-c(outlier_index_vec),]), ncol = 1)
model_residuals_new_df

# Descriptive statistics of the OLS residuals.
basicStats(model_residuals_new_df, ci = 0.95)

# Assignment of weekly index date to all data.
# Define the weekly index dates.
week2 = seq(as.Date("2016-04-03"), length = length(model_residuals_new_df), by = "weeks")

# Plot new residuals.
par(mfrow = c(1,1))
model_residuals_new_df1 = data.frame(model_residuals_new_df, week2)
model_residuals_new_df1$week2 <- as.Date(week2)
require(scales)
```

```
ggplot(model_residuals_new_df1, aes(x = week2, y =  model_residuals_new_df)) +
geom_line(color = "#0B0B0B", size = 2) + xlab("Week, t") +ylab("Residuals, "~epsilon[t])+
geom_hline(yintercept = basicStats(model_residuals_new_df, ci = 0.95)[7,1], linetype = "dashed",
   color = "red", size = 1)+
scale_x_date(labels = date_format("%b %d %Y"), breaks = date_breaks("2 month"))+
scale_y_continuous(limits = c(-2000, 2000), breaks = c(-2000, -1000, 0, 1000, 2000))+
ggsave("time_plot_new.png", height = 3, width = 5)

###################################################
### code chunk number 5:
###################################################
# We then fit different ARMA(p,q)-GARCH(m,n) models to the residuals obtained from the
# the OLS fit.

# Modelling the residuals with an ARMA(0,0)-GARCH(1,1) errors.
model_spec1 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(0,0), include.mean = TRUE), distribution.model = "norm")

#The model fitting.
model_fitted1 = ugarchfit(spec = model_spec1 , data = model_residuals_new_df)
model_fitted1

#----------------------------------------------------------------------------#

# Modelling the residuals with an ARMA(1,0)-GARCH(1,1) errors.
model_spec2 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,0), include.mean = TRUE), distribution.model = "norm")

# The model fitting.
model_fitted2 = ugarchfit(spec = model_spec2 , data = model_residuals_new_df)
model_fitted2

#----------------------------------------------------------------------------#

# Modelling the residuals with an ARMA(0,1)-GARCH(1,1) errors.
model_spec3 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(0,1), include.mean = TRUE), distribution.model = "norm")
```

```
# The model fitting.
model_fitted3 = ugarchfit(spec = model_spec3 , data = model_residuals_new_df)
model_fitted3

#-----------------------------------------------------------------------------#

# Modelling the residuals with an ARMA(1,1)-GARCH(1,1) errors.
model_spec4 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,1), include.mean = TRUE), distribution.model = "norm")

# The model fitting.
model_fitted4 = ugarchfit(spec = model_spec4, data = model_residuals_new_df)
model_fitted4

#-----------------------------------------------------------------------------#

#Modelling the residuals with an ARMA(2,1)-GARCH(1,1) errors.
model_spec5 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,1), include.mean = TRUE), distribution.model = "norm")

#The model fitting.
model_fitted5 = ugarchfit(spec = model_spec5, data = model_residuals_new_df)
model_fitted5

#-----------------------------------------------------------------------------#

# Modelling the residuals with an ARMA(1,2)-GARCH(1,1) errors.
model_spec6 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,2), include.mean = TRUE), distribution.model = "norm")

# The model fitting.
model_fitted6 = ugarchfit(spec = model_spec6, data = model_residuals_new_df)
model_fitted6

#-----------------------------------------------------------------------------#

# Modelling the residuals with an ARMA(2,2)-GARCH(1,1) errors.
model_spec7 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,2), include.mean = TRUE), distribution.model = "norm")

# The model fitting.
model_fitted7 = ugarchfit(spec = model_spec7, data = model_residuals_new_df)
model_fitted7

#------------------------------------------------------------------------------#
# It is determined that from the information criterion, that the best fitted model
# for the OLS model residuals is ARMA(2,2)-GARCH(1,1).
# Modelling the residuals with ARMA(2,2)-GARCH(1,1) errors

#==========================================================#
chosen_residuals_model = model_fitted7
#==========================================================#

###################################################
### code chunk number 6:
###################################################
# Consider a regression model with errors modelled by a ARMA(2,2)-GARCH(1,1) process.
# We will give the results of on simulation run.

# Setting the seed value so as to produce pseudorandom results (i.e. replicatable results).
seed_val = 3216
set.seed(seed_val)
print(seed_val)

#------------------------------------------------------------#
# Intializing vectors for the storage of the beta estimates:
#------------------------------------------------------------#

# Intitialize vectors that will store the beta estimates.
vec_data_beta1_ur = matrix(data = NA, nrow = 1, ncol = 1)
vec_data_beta2_ur = matrix(data = NA, nrow = 1, ncol = 1)


# Vector of OLS model residuals.
#model_residual_vec = matrix(data = c(model_residuals), ncol = 1)
model_residual_vec = model_residuals_new_df
```

```
# Create a matrix storing all the relevant variables being tested.
# The Bitcoin price being the dependent variable and the 2 independent variables being
# the Google searches of Bitcoin and the S&P 500 index.
all_data_data_matrix = matrix(data = cbind(BTC_SP_week[,1], BTC_SP_week[,2], BTC_SP_week[,3]),  ncol = 3)

# We then remove the rows that correpond to the outlier_index_list and use the remaining
# data for our analysis.
all_data_data_matrix_df = data.frame(all_data_data_matrix)
all_data_data_matrix_new_df = all_data_data_matrix_df[-c(outlier_index_vec),]
all_data_data_matrix_new_df



#-------------------------------------------------#
# No splitting and no shuffling:
#-------------------------------------------------#

# Sample size
n = nrow(all_data_data_matrix_new_df)

#----------------------------------#
# Set mean parameters:
#----------------------------------#

# Matrix of the 2 predictor variable observations
X_matrix =  matrix(data =cbind(all_data_data_matrix_new_df[,2], all_data_data_matrix_new_df[,3]),
   nrow = n, ncol = 2)

# Response variable information
Y_T_data =  matrix(data =cbind(all_data_data_matrix_new_df[,1]), nrow = n, ncol = 1)

# Create vectors to store all the iteration of Y_t, epsilon_t
y_t_vec = matrix(data=NA, nrow=n, ncol=1)
epsilon_t_vec = matrix(data = NA, nrow=n, ncol=1)
sigma_t_2_vec = matrix(data = NA, nrow=n, ncol=1)

#----------------------------------#
# Set variance parameters:
```

```
#---------------------------------#
chosen_residuals_model = model_fitted7

residual_model_coefficients = chosen_residuals_model@fit$coef
residual_model_sigma_2 = chosen_residuals_model@fit$var
residual_model_sigma = chosen_residuals_model@fit$sigma

# eta_t  is i.i.d. standard normally distributed (i.e. eta_t ~ N(0,1) for all t).
mean_eta_t  = 0
var_eta_t  = 1
eta_t_1 = rnorm(1 ,mean_eta_t, var_eta_t)
eta_t_2 = rnorm(1 ,mean_eta_t, var_eta_t)

c = residual_model_coefficients[1]
phi_1 = residual_model_coefficients[2]
phi_2 = residual_model_coefficients[3]
theta_1 = residual_model_coefficients[4]
theta_2 = residual_model_coefficients[5]

omega = residual_model_coefficients[6]
alpha_1 = residual_model_coefficients[7]
gamma_1 = residual_model_coefficients[8]

# Note:
# Since s = max(p,q,m,n) = max(2,2,1,1) = 2, which means that there only a need for s=2,
# intial values.
# Therefore epsilon_t_1_squared = epsilon_t_2_squared = sigma_t1_2 = sigma_t2_2 = unconditional_variance.
#model_residuals = matrix(data = model_residuals, ncol = 1)

epsilon_t_1_and_2 = tail(model_residual_vec, n = 2)
epsilon_t_1 = epsilon_t_1_and_2[2]
epsilon_t_2 = epsilon_t_1_and_2[1]

unconditional_variance = sum(epsilon_t_1_and_2**2)

sigma_t1_2 = unconditional_variance
#sigma_t2_2 = unconditional_variance
```

```
sigma_t_2 = unconditional_variance

# Create an empty matrix
matrix_data = matrix(,nrow = n, ncol = 5)

for (t in 1:n){
sigma_t = sqrt(sigma_t_2)

eta_t = rnorm(1, mean_eta_t, var_eta_t)

epsilon_t = c + ((phi_1)*(epsilon_t_1)) +((phi_2)*(epsilon_t_2)) + ((theta_1)*(eta_t_1)) +
((theta_2)*(eta_t_2)) + ((sigma_t)*(eta_t))

sigma_t_2 = omega + ((alpha_1)*(epsilon_t_1**2)) + ((gamma_1)*(sigma_t1_2))

if (t > 0){
y_t = Y_T_data[t,1]
x_1 = X_matrix[t,1]
x_2 = X_matrix[t,2]
epsilon_t_2 = epsilon_t_1
epsilon_t_1 = epsilon_t
eta_t_2 = eta_t_1
eta_t_1 = eta_t
sigma_t1_2 = sigma_t_2
}
y_t_vec[[t]] = y_t
epsilon_t_vec[[t]] = epsilon_t
sigma_t_2_vec[[t]] = sigma_t_2
print(paste("t:", t, "Y_T", y_t, "epsilon is :", epsilon_t, "Var_epsilont is :", sigma_t_2))
}

matrix_data = matrix(data = cbind(y_t_vec,X_matrix[,1],X_matrix[,2],epsilon_t_vec, sigma_t_2_vec),
 nrow = n, ncol = 5)
print(head(matrix_data))

# Create weekly indicators from 03 April 2016.
week = seq(as.Date("2016-04-03"), length = length(y_t_vec), by = "weeks")
```

```
# Create a data frame with all the weekly data.
all_data = data.frame(matrix_data)

# The assignment of the weekly index date to the new data frame and stored in the DataFrame.
new_fitted_results = xts(all_data, week)

# Print the data to view.
head(new_fitted_results)

# Descriptive statistics of the new data frame.
basicStats(new_fitted_results, ci = 0.95)

# Plot y_t, epsilon_t, sigma_t_2
par(mfrow = c(3,1))
plot(new_fitted_results[,1], type = "l", ylab = "Y_t", xlab = "Date", main = " ")
plot(new_fitted_results[,4], type = "l", ylab = "epsilon_t", xlab = "Date", main = " ")
plot(new_fitted_results[,5], type = "l", ylab = "sigma_t_2", xlab = "Date", main = " ")

new_fitted_results_matrix = matrix(data = new_fitted_results, ncol = 5)

# Log-likelihood function.
logLikFun = function(beta_1, beta_2) {
T = length(new_fitted_results_matrix[,1])
#beta_1 = param[1]
#beta_2 = param[2]

mu = mean(new_fitted_results_matrix[,4])

epsilon_tt = new_fitted_results_matrix[,1] - (((beta_1)*(new_fitted_results_matrix[,2])) + ((beta_2)*(new_
sigma_t_22 = new_fitted_results_matrix[,5]

ll = -((-0.5*T*log(2*pi)) - ((0.5)*sum(log(sigma_t_2))) - ((0.5)*(sum(((epsilon_tt-mu)**2)/(sigma_t_22))))

}
# Maximum log-likelihood estimation.
#mle_trail = maxLik(logLik = logLikFun, start = c(beta_1 = -500, beta_2 = -500))
mle_trail = mle(minuslogl = logLikFun, start = list(beta_1 = 0, beta_2 = 0), method = "BFGS")
print(summary(mle_trail))
```

```
# Unrestricted regression coefficients estimated from the model.
estimate_beta1 = coef(mle_trail)[1]
estimate_beta2 = coef(mle_trail)[2]
vec_data_beta1_ur = estimate_beta1
vec_data_beta2_ur = estimate_beta2

matrix_data_beta_ur_sim_one = matrix(data =c(vec_data_beta1_ur, vec_data_beta2_ur), ncol = 2)
print(matrix_data_beta_ur_sim_one)

#****************************************************#
# The estimates produced will be our true estimates.

true_parameter_estimates = matrix_data_beta_ur_sim_one
print(true_parameter_estimates)

#*****************************************************#

###################################################
### code chunk number 7:
###################################################
# In the iterative process we will estimate each of the regression coefficients,
# N times.
# We will use the fitted ARMA(1,1)-GARCH(1,1) estimates of the residuals in
# order to estimate the regression coefficients.

#---------------------------------#
# N simulation run:
#---------------------------------#
# Number of simulations.
N_sim = 10000

#-----------------------------------------------------------#
# Intializing vectors for the storage of the beta estimates:
#-----------------------------------------------------------#

# Intitialize vectors that will store the beta estimates.
# We have the true beta estimates.
```

```
true_beta_1 = true_parameter_estimates[,1]

true_beta_2 = true_parameter_estimates[,2]

vec_data_beta1_true = matrix(data = true_beta_1, nrow = N_sim, ncol = 1)

vec_data_beta2_true = matrix(data = true_beta_2, nrow = N_sim, ncol = 1)


vec_data_beta1_ur = matrix(data = NA, nrow = N_sim, ncol = 1)

vec_data_beta2_ur = matrix(data = NA, nrow = N_sim, ncol = 1)



for (i in 1:N_sim){

# Set the seed value which changes with each iteration.

seed_val = 455 + i

set.seed(seed_val)

print(seed_val)


# Vector of OLS model residuals.

model_residual_vec = model_residuals_new_df


# Create a matrix storing all the relevant variables being tested.

# The Bitcoin price being the dependent variable and the 2 independent variables being

# the Google searches of Bitcoin and the S&P 500 index.

all_data_data_matrix = matrix(data = cbind(all_data_data_matrix_new_df[,1],

   all_data_data_matrix_new_df[,2],

   all_data_data_matrix_new_df[,3]),  ncol = 3)


#------------------------------------------------#

# Splitting data into training and test sets:

#------------------------------------------------#

# Split data into one 70/30 split - 70% Training and 30% Testing.


# First we reshuffled the information.

shuffled_rows = sample(nrow(all_data_data_matrix), replace = TRUE)

shuffled_BTC_SP_week = all_data_data_matrix[shuffled_rows,]

shuffled_model_residuals = model_residual_vec[shuffled_rows,]


# Now we can do the splitting.

split = round(nrow(shuffled_BTC_SP_week)*0.70)

train_shuffled_BTC_SP_week = shuffled_BTC_SP_week[1:split, ]
```

```
test_shuffled_BTC_SP_week = shuffled_BTC_SP_week[(split+1):nrow(shuffled_BTC_SP_week), ]

shuffled_model_residuals = matrix(data = shuffled_model_residuals, ncol = 1)
split1 = round(nrow(shuffled_model_residuals)*0.70)
train_shuffled_model_residuals = matrix(data = shuffled_model_residuals[1:split1, ], ncol = 1)
test_shuffled_model_residuals = matrix(data =
 shuffled_model_residuals[(split1+1):nrow(shuffled_model_residuals), ], ncol = 1)

# Sample size
#n = nrow(all_data_data[,1])
n = nrow(train_shuffled_BTC_SP_week)

#---------------------------------#
# Set mean parameters:
#---------------------------------#

# Matrix of the 2 predictor variable observations
X_matrix =  matrix(data =cbind(train_shuffled_BTC_SP_week[,2], train_shuffled_BTC_SP_week[,3]),
   nrow = n, ncol = 2)

# Response variable information
Y_T_data =  matrix(data =c(train_shuffled_BTC_SP_week[,1]), nrow = n, ncol = 1)

# Create vectors to store all the iteration of Y_t, epsilon_t
y_t_vec = matrix(data=NA, nrow=n, ncol=1)
epsilon_t_vec = matrix(data=NA, nrow=n, ncol=1)
sigma_t_2_vec = matrix(data=NA, nrow=n, ncol=1)

#---------------------------------#
# Set variance parameters:
#---------------------------------#
chosen_residuals_model = model_fitted7

residual_model_coefficients = chosen_residuals_model@fit$coef
residual_model_sigma_2 = chosen_residuals_model@fit$var
residual_model_sigma = chosen_residuals_model@fit$sigma

# eta_t  is i.i.d. standard normally distributed (i.e. eta_t ~ N(0,1) for all t).
```

```
mean_eta_t  = 0

var_eta_t  = 1

eta_t_1 = rnorm(1 ,mean_eta_t, var_eta_t)

eta_t_2 = rnorm(1 ,mean_eta_t, var_eta_t)



c = residual_model_coefficients[1]

phi_1 = residual_model_coefficients[2]

phi_2 = residual_model_coefficients[3]

theta_1 = residual_model_coefficients[4]

theta_2 = residual_model_coefficients[5]


omega = residual_model_coefficients[6]

alpha_1 = residual_model_coefficients[7]

gamma_1 = residual_model_coefficients[8]


# Note:

# Since s = max(p,q,m,n) = max(1,1,1,1) = 1, which means that there only a need for s=1,

# intial values.

# Therefore epsilon_t_1_squared = sigma_t1_2 = unconditional_variance.

epsilon_t_1_and_2 = tail(train_shuffled_model_residuals, n = 2)

epsilon_t_1 = epsilon_t_1_and_2[2]

epsilon_t_2 = epsilon_t_1_and_2[1]


unconditional_variance = sum(epsilon_t_1_and_2**2)


sigma_t1_2 = unconditional_variance

#sigma_t2_2 = unconditional_variance


sigma_t_2 = unconditional_variance


# Create an empty matrix

matrix_data = matrix(,nrow = n, ncol = 5)


for (t in 1:n){

sigma_t = sqrt(sigma_t_2)


eta_t = rnorm(1, mean_eta_t, var_eta_t)
epsilon_t = c + ((phi_1)*(epsilon_t_1)) +((phi_2)*(epsilon_t_2)) + ((theta_1)*(eta_t_1)) +
((theta_2)*(eta_t_2)) + ((sigma_t)*(eta_t))

sigma_t_2 = omega + ((alpha_1)*(epsilon_t_1**2)) + ((gamma_1)*(sigma_t1_2))

if (t > 0){
y_t = Y_T_data[t,1]
x_1 = X_matrix[t,1]
x_2 = X_matrix[t,2]
epsilon_t_2 = epsilon_t_1
epsilon_t_1 = epsilon_t
eta_t_2 = eta_t_1
eta_t_1 = eta_t
sigma_t1_2 = sigma_t_2
}

y_t_vec[[t]] = y_t
epsilon_t_vec[[t]] = epsilon_t
sigma_t_2_vec[[t]] = sigma_t_2
print(paste("t:", t, "Y_T", y_t, "epsilon is :", epsilon_t, "Var_epsilont is :", sigma_t_2))

}

matrix_data = matrix(data =cbind(y_t_vec,X_matrix[,1],X_matrix[,2],epsilon_t_vec, sigma_t_2_vec),
 nrow = n, ncol = 5)
print(head(matrix_data))

# Create weekly indicators from 03 April 2016.
week = seq(as.Date("2016-04-03"), length = length(y_t_vec), by = "weeks")

# Create a data frame with all the weekly data
all_data = data.frame(matrix_data)

# The assignment of the weekly index date to the new data frame and stored in the DataFrame.
new_fitted_results = xts(all_data, week)

# Print the data to view it.
```

```
head(new_fitted_results)

# Descriptive statistics of the new data frame.
basicStats(new_fitted_results, ci = 0.95)

# Plot y_t, epsilon_t, sigma_t_2.
par(mfrow = c(3,1))
plot(new_fitted_results[,1], type = "l", ylab = "Y_t", xlab = "Date", main = " ")
plot(new_fitted_results[,4], type = "l", ylab = "epsilon_t", xlab = "Date", main = " ")
plot(new_fitted_results[,5], type = "l", ylab = "sigma_t_2", xlab = "Date", main = " ")

new_fitted_results_matrix = matrix(data = new_fitted_results, ncol = 5)

# Log-likelihood function.
logLikFun = function(beta_1, beta_2){
T = length(new_fitted_results_matrix[,1])
#beta_1 = param[1]
#beta_2 = param[2]

mu = mean(new_fitted_results_matrix[,4])

epsilon_tt = new_fitted_results_matrix[,1] - (((beta_1)*(new_fitted_results_matrix[,2]))+
((beta_2)*(new_fitted_results_matrix[,3])))
sigma_t_22 = new_fitted_results_matrix[,5]

ll = -((-0.5*T*log(2*pi))-((0.5)*sum(log(sigma_t_2))) - ((0.5)*(sum(((epsilon_tt-mu)**2)/(sigma_t_22)))))

}
# Maximum log-likelihood estimation (MLE).
#mle_trail = maxLik(logLik = logLikFun, start = c(beta_1 = 0, beta_2 = 0))
mle_trail = mle(minuslogl = logLikFun, start = list(beta_1 = 0, beta_2 = 0), method = "BFGS")
print(summary(mle_trail))

# Unrestricted regression coefficients estimated from the model.
estimate_beta1 = coef(mle_trail)[1]
estimate_beta2 = coef(mle_trail)[2]
vec_data_beta1_ur[[i]] = estimate_beta1
vec_data_beta2_ur[[i]] = estimate_beta2
```

```
matrix_data_beta = matrix(data =cbind(estimate_beta1, estimate_beta2), ncol = 2)
print(head(matrix_data_beta))

}

# All the estimates after N_sim iterations.
matrix_data_beta_ur_model = matrix(data =cbind(vec_data_beta1_ur, vec_data_beta2_ur),
   nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_ur_model))

# Descriptive statistics of the beta estimates.
basicStats(matrix_data_beta_ur_model, ci = 0.95)

# Check for any error warnings.
warnings()

#------------------------------------------------------------------------------------------#
# Range of the unrestricted regression coefficient estimates.
# Range of Beta_1 unrestricted estimates.
beta1_ur_range = basicStats(matrix_data_beta_ur_model, ci = 0.95)[4,1] -
 basicStats(matrix_data_beta_ur_model, ci = 0.95)[3,1]
beta1_ur_range


# Range of Beta_2 unrestricted estimates.
beta2_ur_range = basicStats(matrix_data_beta_ur_model, ci = 0.95)[4,2] -
basicStats(matrix_data_beta_ur_model, ci = 0.95)[3,2]
beta2_ur_range

#------------------------------------------------------------------------------------------#
# Interquartile range (IQR) is a measure of variability.
# IQR = Q3-Q1
# IQR of Beta_1 unrestricted estimates.
beta1_ur_IQR = basicStats(matrix_data_beta_ur_model, ci = 0.95)[6,1] -
  basicStats(matrix_data_beta_ur_model, ci = 0.95)[5,1]
beta1_ur_IQR
```

```
# IQR of Beta_2 unrestricted estimates.
beta2_ur_IQR = basicStats(matrix_data_beta_ur_model, ci = 0.95)[6,2] -
  basicStats(matrix_data_beta_ur_model, ci = 0.95)[5,2]
beta2_ur_IQR

#-------------------------------------------------------------------------------------------#

# Plot histograms of each of the beta estimate's distribution.
# Histogram of Beta_1 distribution.
par(mfrow = c(1,1))
xfit1 = seq(min(matrix_data_beta_ur_model[,1]), max(matrix_data_beta_ur_model[,1]),
length = length(matrix_data_beta_ur_model[,1]))
yfit1 = dnorm(xfit1, mean = mean(matrix_data_beta_ur_model[,1]),
  sd = sd(matrix_data_beta_ur_model[,1]))

# Histogram of the unrestricted Beta_1 estimates distribution.
hist(matrix_data_beta_ur_model[,1], nclass = 100, probability = TRUE, xlab = expression(beta[1]),
 main = "")

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit1, yfit1, lwd = 2, col = "red")

#-------------------------------------------------------------------------------------------#

# Histogram of Beta_2 distribution.
par(mfrow = c(1,1))
xfit2 = seq(min(matrix_data_beta_ur_model[,2]), max(matrix_data_beta_ur_model[,2]),
length = length(matrix_data_beta_ur_model[,2]))
yfit2 = dnorm(xfit2, mean = mean(matrix_data_beta_ur_model[,2]),
  sd = sd(matrix_data_beta_ur_model[,2]))

# Histogram of the unrestricted Beta_2 estimates distribution.
hist(matrix_data_beta_ur_model[,2], nclass = 100, probability = TRUE, xlab = expression(beta[2]),
 main = "")

# Fitting a normal distribution curve with the actual true values of the data.
lines(xfit2, yfit2, lwd = 2, col = "purple")
```

```
###################################################
### code chunk number 8:
###################################################
# The preliminary test.
# Consider the N bootstrapped samples that were used to estimate the regression coefficients
# for the unrestricted model.
# We will then determine the 100(1-alpha)% confidence interval of each regression coefficient
# estimate for different levels of significance and  then determine whether the restricted or
# unrestricted model will be chosen for each case.
# If zero is found within the confidence interval of Beta_2, then we do not reject the null
# hypothesis (i.e. restricted model chosen over the unrestricted model).
# pi values grid.

# Confidence intervals of for different alpha values (i.e. level of significance).
# alpha values grid values.
alpha_values = c(0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5)

# (1-alpha_values) values grid.
one_minus_alpha_values = 1-alpha_values

# Create an empty matrix for the confidence interval limits of the unrestricted beta estimate values.
CI_ll_ul_limits_beta_1_ur = matrix(data = NA , nrow = length(alpha_values), ncol = 3)
CI_ll_ul_limits_beta_2_ur = matrix(data = NA , nrow = length(alpha_values), ncol = 3)

# Compute the confidence intervals.
for (i in 1:length(alpha_values)){
# The mean values of the unrestricted parameter estimates.
mean_beta_1_ur = basicStats(matrix_data_beta_ur_model, ci = one_minus_alpha_values[[i]])[7,1]
mean_beta_2_ur = basicStats(matrix_data_beta_ur_model, ci = one_minus_alpha_values[[i]])[7,2]
print(mean_beta_2_ur)

# The standard deviation values of the unrestricted parameter estimates.
sd_beta_1_ur = basicStats(matrix_data_beta_ur_model, ci = one_minus_alpha_values[[i]])[14,1]
sd_beta_2_ur = basicStats(matrix_data_beta_ur_model, ci = one_minus_alpha_values[[i]])[14,2]
print(sd_beta_2_ur)

# The 100(1-alpha/2)quantile values of the unrestricted parameter estimates.
quantile_value = 1 - (alpha_values[[i]]/2)
```

```
z_alpha_div_by_2 = qnorm(quantile_value)

# The margins of error values of the unrestricted parameter estimates.
moe_beta_1_ur = z_alpha_div_by_2*(sd_beta_1_ur/(sqrt(N_sim)))
moe_beta_2_ur = z_alpha_div_by_2*(sd_beta_2_ur/(sqrt(N_sim)))
print(moe_beta_2_ur)

# Lower and upper confidence limits for Beta_1
CI_ll_ul_limits_beta_1_ur[i,1] = one_minus_alpha_values[[i]]*100
CI_ll_ul_limits_beta_1_ur[i,2] = mean_beta_1_ur - moe_beta_1_ur
CI_ll_ul_limits_beta_1_ur[i,3] = mean_beta_1_ur + moe_beta_1_ur

# Lower and upper confidence limits for Beta_2
CI_ll_ul_limits_beta_2_ur[i,1] = one_minus_alpha_values[[i]]*100
CI_ll_ul_limits_beta_2_ur[i,2] = mean_beta_2_ur - moe_beta_2_ur
CI_ll_ul_limits_beta_2_ur[i,3] = mean_beta_2_ur + moe_beta_2_ur

}

# Print the confidence interval limits of the unrestricted beta estimate values.
print(CI_ll_ul_limits_beta_1_ur)
print(CI_ll_ul_limits_beta_2_ur)

###################################################
### code chunk number 9:
###################################################
#-------------------------------------------------#
# Matrices used to compute the MSEs:
#-------------------------------------------------#
# We demonstrate linear shrinakage estimation on the real data.

# Matrix of true beta estimates.
# We have the true beta estimates.
true_beta_1 = matrix_data_beta_ur_sim_one[,1]
true_beta_2 = matrix_data_beta_ur_sim_one[,2]
vec_data_beta1_true = matrix(data = true_beta_1, nrow = N_sim, ncol = 1)
vec_data_beta2_true = matrix(data = true_beta_2, nrow = N_sim, ncol = 1)
matrix_data_beta_true0 = matrix(data = cbind(vec_data_beta1_true, vec_data_beta2_true),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_true0))



# Matrix of unrestricted beta estimates.
matrix_data_beta_ur = matrix(data = cbind(vec_data_beta1_ur, vec_data_beta2_ur),
 nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_ur))

# We have the restricted beta estimates.
vec_data_beta1_r = matrix(data = cbind(vec_data_beta1_ur), nrow = N_sim, ncol = 1)
vec_data_beta2_r = matrix(data = 0, nrow = N_sim, ncol = 1)
# Matrix of restricted estimators to be tested.
matrix_data_beta_r = matrix(data = cbind(vec_data_beta1_r, vec_data_beta2_r),
nrow = N_sim, ncol = 2)
print(head(matrix_data_beta_r))

#----------------------------------------------#
# Computing the shrinkage estimates:
#----------------------------------------------#
# pi values grid.
pi_values = c(0, 0.01, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,
  0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1)

# (1-pi) values grid.
one_minus_pi_values = 1-pi_values

# Create an empty matrix for the beta shrinkage estimates.
matrix_data_beta_shrinkage = matrix(data = NA, nrow = N_sim, ncol = 2)

# Create an empty matrix for the beta shrinkage estimate MSE values.
MSE_pi_values= matrix(data = NA , nrow = length(pi_values), ncol = 3)

# Compute the MSE values.
for (i in 1:length(pi_values)){
matrix_data_beta_shrinkage[,1] = ((pi_values[[i]]*matrix_data_beta_ur[,1])+
  (one_minus_pi_values[[i]]*matrix_data_beta_r[,1]))
matrix_data_beta_shrinkage[,2] = ((pi_values[[i]]*matrix_data_beta_ur[,2])+
      (one_minus_pi_values[[i]]*matrix_data_beta_r[,2]))
MSE_pi_values[i,1] = pi_values[[i]]
MSE_pi_values[i,2] = one_minus_pi_values[[i]]
MSE_pi_values[i,3] = (1/N_sim)*
 (sum((t(matrix_data_beta_shrinkage-matrix_data_beta_true0))%*%
 (matrix_data_beta_shrinkage-matrix_data_beta_true0)))
}

# Print the MSE values for each pi value chosen.
print(MSE_pi_values)

#**************************************************#
#---------------------- End ----------------------#
#**************************************************#
```

Bibliography

[1] K. Aas and I. H. Haff. The generalized hyperbolic skew Student-t distribution. *Journal of Financial Econometrics*, 4(2):275–309, 2006.

[2] A. Aknouche and A. Bibi. Quasi-maximum likelihood estimation of periodic GARCH and periodic ARMA-GARCH processes. *Journal of Time Series Analysis*, 30(1):19–46, 2009.

[3] L. Bachelier. Theory of speculation (1900). *The Random Character of Stock Market Prices*, 4(1):91–193, 1964.

[4] T. A. Bancroft. On biases in estimation due to the use of preliminary tests of significance. *The Annals of Mathematical Statistics*, 15(2):190–204, 1944.

[5] O. Barndorff-Nielsen. Exponentially decreasing distributions for the logarithm of particle size. *Proceedings of the Royal Society of London A*, 353(1674):401–419, 1977.

[6] O. E. Barndorff-Nielsen and P. Blæsild. Hyperbolic distributions and ramifications: contributions to theory and application. In C. Taillie, P. Patil, and B. A. Baldessari, editors, *Statistical distributions in scientific work*, volume 4, pages 19–44. Springer, 1981.

[7] A. K. Bera and M. L. Higgins. ARCH models: properties, estimation and testing. *Journal of Economic Surveys*, 7(4):305–366, 1993.

[8] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[9] T. Bollerslev. A conditionally heteroskedastic time series model for speculative prices and rates of return. *The Review of Economics and Statistics*, 69(3):542–547, 1987.

[10] J. Bouoiyour, R. Selmi, A. K. Tiwari, O. R. Olayeni, et al. What drives Bitcoin price? *Economics Bulletin*, 36(2):843–850, 2016.

[11] G. E. P. Box and G. M. Jenkins. *Time series analysis*, volume 1. John Wiley & Sons, 1970.

[12] J. Chu, S. Chan, S. Nadarajah, and J. Osterrieder. GARCH modelling of cryptocurrencies. *Journal of Risk and Financial Management*, 10(17):1–15, 2017.

[13] J. Chu, S. Nadarajah, and S. Chan. Statistical analysis of the exchange rate of Bitcoin. *PLoS ONE*, 10(7):e0133678, 2015.

[14] R Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2018.

[15] A. de Moivre. The doctrine of chances. In *Annotated Readings in the History of Statistics*, pages 32–36. Springer, 2001.

[16] R. Engle, D. M. Lilien, and R. P. Robins. Estimating time-varying risk premia in the term structure the ARCH-M model. *Econometrica: Journal of the Econometric Society*, 55(2):391–407, 1987.

[17] R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007, 1982.

[18] C. Fernández and M. F. J. Steel. On Bayesian modeling of fat tails and skewness. *Journal of the American Statistical Association*, 93(441):359–371, 1998.

[19] C. Francq and J. M. Zakoian. Maximum likelihood estimation of pure GARCH and ARMA-GARCH processes. *Bernoulli*, 10(4):605–637, 2004.

[20] C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, volume 7. Perthes et Besser, 1809.

[21] I. Gavrilov and R. Pusev. *normtest: Tests for Normality*, 2014. R package version 1.1.

[22] A. Ghalanos. *rugarch: Univariate GARCH models.*, 2018. R package version 1.4-0.

[23] L. R. Glosten, R. Jagannathan, and D. E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48(5):1779–1801, 1993.

[24] W. S. Gösset. The probable error of a mean. *Biometrika*, 6:1–25, 1908.

[25] C. W. J. Granger and R. Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, 1(1):15–29, 1980.

[26] J. D. Hamilton. *Time Series Analysis*, volume 2. Princeton University Press, USA, 1994.

[27] A. Henningsen and O. Toomet. maxLik: A package for maximum likelihood estimation in R. *Computational Statistics*, 26(3):443–458, 2011.

[28] J. R. M. Hosking. Fractional differencing. *Biometrika*, 68(1):165–176, 1981.

[29] S. Hossain and M. Ghahramani. Shrinkage estimation of linear regression models with GARCH errors. *Journal of Statistical Theory and Applications*, 15(4):405–423, 2016.

[30] N. L. Johnson. Systems of frequency curves generated by methods of translation. *Biometrika*, 36(1/2):149–176, 1949.

[31] P. Katsiampa. Volatility estimation for Bitcoin: a comparison of GARCH models. *Economics Letters*, 158:3–6, 2017.

[32] B. Klein. The demand for quality-adjusted cash balances: price uncertainty in the US demand for money fun. *Journal of Political Economy*, 85:691–715, 1977.

[33] S.-W. Lee and B. E. Hansen. Asymptotic theory for the GARCH (1, 1) quasi-maximum likelihood estimator. *Econometric Theory*, 10(1):29–52, 1994.

[34] D. Luethi and W. Breymann. *ghyp: A package on Generalized Hyperbolic Distribution and its special cases*, 2016. R package version 1.5.7.

[35] R. L. Lumsdaine. Asymptotic properties of quasi-maximum likelihood estimator in GARCH (1, 1) and IGARCH (II) models. *Unpublished Manuscript, Princeton University and National Bureau of Economic Research*, 1991.

[36] B. B. Mandelbrot. The variation of certain speculative prices. *The Journal of Business*, 36:394–394, 1963.

[37] S. B. Messaoud and C. Aloui. Measuring risk of portfolio: GARCH-copula model. *Journal of Economic Integration*, 30(1):172–205, March 2015.

[38] D. B. Nelson. Conditional heteroskedasticity in asset returns: a new approach. *Econometrica: Journal of the Econometric Society*, pages 347–370, 1991.

[39] B. Pfaff and A. McNeil. *QRM: Provides R-language code to examine quantitative risk management concepts*, 2016. R package version 0.4-13.

[40] M. Polasik, A. I. Piotrowska, T. P. Wisniewski, R. Kotkowski, and G. Lightfoot. Price fluctuations and the use of Bitcoin: an empirical inquiry. *International Journal of Electronic Commerce*, 20(1):9–49, 2015.

[41] K. Prause. *The Generalized Hyperbolic model: estimation, financial derivatives and risk measures.* PhD thesis, Verlag nicht ermittelbar, 1999.

[42] R. Rabemananjara and J. M. Zakoian. Threshold ARCH models and asymmetries in volatility. *Journal of Applied Econometrics*, 8(1):31–49, 1993.

[43] J. S. Rao and S. Debasis. *Linear models: an integrated approach*, volume 6. World Scientific, 2003.

[44] R. Reider. Volatility forecasting I: GARCH models. *New York*, 2009.

[45] R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

[46] J. A. Ryan and J. M. Ulrich. *quantmod: Quantitative Financial Modelling Framework*, 2019. R package version 0.4-15.

[47] A. K. Md. E. Saleh. *Theory of preliminary test and Stein-type estimation with applications*, volume 517. John Wiley & Sons, 2006.

[48] A. K. Md. E. Saleh, M. Arashi, and B. M. Golam Kibria. *Theory of ridge regression estimation with applications*, volume 285. John Wiley & Sons, USA, 2019.

[49] A. K. Md. E. Saleh, M. Arashi, and S. M. M. Tabatabaey. *Statistical inference for models with multivariate t-distributed errors*. John Wiley & Sons, Inc, New Jersey, 2014.

[50] D. J. Scott, D. Würtz, C. Dong, and T. T. Tran. Moments of the generalized hyperbolic distribution. *Computational Statistics*, 26(3):459–476, 2011.

[51] Y. Sovbetov. Factors influencing cryptocurrency prices: evidence from Bitcoin, Ethereum, Dash, Litcoin, and Monero. *Journal of Economics and Financial Analysis*, 2(2):1–27, 2018.

[52] M. T. Subbotin. On the law of frequency of error. *Mathematics Collection*, 31(2):296–301, 1923.

[53] trends.google.com. Google Trends, 2019. Online; accessed 19-July-2019.

[54] R. S. Tsay. *Analysis of Financial Time Series*, volume 543. John Wiley & Sons, 2010.

[55] F. Van den Bossche, G. Wets, and T. Brijs. A regression model with ARIMA errors to investigate the frequency and severity of road traffic accidents. Steunpunt Verkeersveiligheid, 2004.

[56] J. Vent. *crypto: Cryptocurrency market data*, 2018. R package version 1.0.2.

[57] N. Y. Vo and G. Xu. The volatility of Bitcoin returns and its correlation to financial markets. In *2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC)*, pages 1–6. IEEE, 2017.

[58] G. Walker. On periodicity in series of related terms. *Proceedings of the Royal Society of London*, 131(818):518–532, 1931.

[59] A. A. Weiss. Asymptotic theory for ARCH models: estimation and testing. *Econometric Theory*, 2(1):107–131, 1986.

[60] P. Whittle. *Hypothesis testing in time series analysis*. PhD thesis, Uppsala University, 1951. Almquist and Wiksell, Uppsala.

[61] H. Wold. Causality and econometrics. *Econometrica: Journal of the Econometric Society*, pages 162–177, 1954.

[62] D. Wuertz, T. Setz, Y. Chalabi, C. Boudt, P. Chausse, and M. Miklovac. *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*, 2017. R package version 3042.83.

[63] D. Yermack. Is Bitcoin a real currency? An economic appraisal. In *Handbook of digital currency*, pages 31–43. Elsevier, 2015.

[64] G. U. Yule. On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London Series A*, 226:267–298, 1927.

[65] J. M. Zakoian. Threshold heteroskedastic models. *Journal of Economic Dynamics and Control*, 18(5):931–955, 1994.