LUISS

**Department of Economics and Finance**

Chair of Asset Pricing

# Forecasting Bitcoin Time Series with ARIMA-GARCH and Recurrent Neural Networks

Prof. Nicola Borri

SUPERVISOR

Prof. Pietro Reichlin

CO-SUPERVISOR

Daniele Cuomo - Matr.713801

CANDIDATE

Academic Year 2019/2020

# Contents

# Introduction

The main objective of this thesis is to provide a comprehensive description of the pioneer of the cryptocurrency revolution, Bitcoin, and perform a comparative analysis between the classic econometric and the modern machine learning approaches in modeling time series and forecasting the price of a financial asset, while accounting for the theoretical background of both methods to interpret and understand their functioning.

Chapter 1 provides an overview of the technical characteristic of the Bitcoin technology, the basic concepts of cryptography, hash functions and decentralization, by which Bitcoin draws its innovative and disruptive features that, throughout the years, attracted the attention of tech enthusiasts and investors in financial markets, setting the cornerstone for an unprecedented technological revolution. After describing how the blockchain works and how its application could be beneficial in many use cases, I reported the main criticisms about Bitcoin and considered the future outlooks for the whole cryptocurrency sector. Finally, I analyzed the Bitcoin economics, comparing it to more traditional store of value like gold, its state of adoption, if and why it should be considered a bubble and finally compared its returns in the last years with respect to other financial assets.

In Chapter 2, after outlining the theoretical framework of the Box and Jenkins approach to time series modeling and the Engle and Bollerslev models for conditional heteroscedasticity, I downloaded the historical 4 hour close prices of BTC/USDT through Python, calling the API of Binance exchange, and performed a statistical and econometric analysis of the data set with R. I started to fit a simple ARIMA model to the log-transformed time series of Bitcoin and then augmented it with GARCH, with the aim of finding if the extension for conditional volatility can provide an adequate approximation of the high volatile price dynamics of BTC and if the out-of-sample forecasts could contribute in enhancing a speculative trading strategy.

Finally, in Chapter 3, I explored the cutting-edge architectures of recurrent neural networks by describing the path from the first artificial neuron, to the complex sequential learning models, passing through an overview of state of the art learning and optimization algorithms that made possible to apply deep learning techniques in real life problems and made neural networks popular in every filed, including the financial sector. With *Scikit-learn*, *Keras* and *Tensorflow* Python libraries, I trained an $\alpha$-RNN model with exponentially weighted hidden states and analyzed the final results, comparing the out-of-sample performance with respect to the econometric model specified in the earlier chapter, indicating the potentialities of the model and how it could be tweaked to obtain a more accurate approximation of the non-linear patterns in the Bitcoin time series.

# Chapter 1

# The Bitcoin Technology

## 1.1 History of Bitcoin

### 1.1.1 A Brief Overview

When in late 2008, an unknown person, or perhaps a group of people, under the pseudonym of Satoshi Nakamoto posted on the website *bitcoin.org* a white paper called "Bitcoin: a Peer-to-Peer electronic Cash System" [1], presenting an innovative solution of a trustless and decentralized payment system, the first cryptocurrency was set to be the cornerstone for a new type of digital asset, paving the way to the creation of thousands of other different cryptocurrencies in subsequent years.

Gradually but consistently gaining the attention of multiple institutions, ranging from the financial to the technology sectors, and of numerous individuals such as academics, researchers and investors, implications and effects of cryptocurrecies on the global economy have been under scrutiny, with the main concern about the potential risks that they could bring to the financial system, not to mention the benefits that the underlying technology, the blockchain, could bring not only to financial companies but to society as a whole.

While the identity of Nakamoto is still unknown, one might focus on his disruptive yet inspiring idea, a decentralized network that would allow its participants to send and receive digital currency directly, without the need of an intermediary. As a matter of fact, the modern payment systems are built around the need of a trusted central party, like a bank, to ensure that the transaction between the payer and the payee goes through, bearing all the operations needed to transfer the designated amount of money between them, like for example in cross-border payments. One of the main concerns of Nakamoto was to overcome the intrinsic limitations of the modern, centralized, payment system, namely the cost of mediation that would hinder the possibility of using electronic payments for small casual transactions and the limited feasibility of making non-reversible payments for non-reversible services, as financial institutions often need to mediate disputes between parties of a transaction.

The role of the trusted third party is then completely replaced by a cryptographic algorithm that builds confidence between the participants of the network, acting as a proof that the transactions are genuine and the funds transferred exists, avoiding Bitcoins counterfeiting, a problem called *double-spending*. All the transactions between the peers are immutably recorded in blocks, and every new transaction points back to previous transactions contained in earlier blocks, thus forming a chain of blocks. This chain, called the blockchain, is the ledger where the information about the entire history of all the transactions that ever happened in the network is recorded on. The peculiarity of this electronic ledger lies in its distributed nature as anyone can become a 'node' of the network by downloading the open-source bitcoin client and start participating in the network. The creation of a bitcoin address is needed to start transferring funds in a peer-to-peer fashion.

Additionally, some nodes will act as *miners*, participants that make the computing power at their disposal available to the network in order to compete in the creation of new blocks, that is, wrapping new pending transactions up and earning both a fee for their effort and newly minted bitcoins when they manage to create a new block before everyone else. A more technical overview about the bitcoin and its underlying technologies is presented in the next sections.

### 1.1.2 From Inception to Modern Days

Bitcoin was officially implemented in January 2009, and the Bitcoin network was created in the same month after Nakamoto successfully mined the first block, called the *genesis block*. The *coinbase* [1] contained a message quoting the following headline of *The Times* newspaper: "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks", which functioned as a timestamp, proving that the block could not have been mined beforehand, and was also interpreted as a criticism. Shortly thereafter, the first transaction on the Bitcoin network was made from Satoshi to cypherpunk[2] and cryptographer Hal Finney. Numerous leading figures in the cypherpunk community started supporting Bitcoin, like Wei Dai, the creator of b-money which is considered to be the predecessor of modern cryptocurrencies, and Nick Szabo, inventor of bit gold. In 2010 Laszlo Hanyecz bought two pizzas for ₿10,000, which are now worth more than 200 million USD, this was the first successful commercial transaction made with Bitcoin. Before disappearing, Nakamoto mined approximately one million bitcoins and left the control of the code repository to Gavin Andersen, who later became the lead developer at the Bitcoin Foundation, a nonprofit organization founded in 2012 and centered around the development and promotion of Bitcoin.

In 2010, Jed McCaleb founded Mt Gox, which was the largest, most well-known and used Bitcoin exchange. Mt Gox would eventually go bankrupt in 2014 due to a number of reasons and

[1] It is the very first transaction in a new block and of a special kind since it indicates the amount of new bitcoins that is to be awarded to the miner who successfully created the block.

[2] An activist engaged in the homonym movement started in 1980s, advocating and supporting privacy-enhancing technologies heavily backed and secured by cryptography, as the foundation for a socio-political change.

scandals, such as a security breach which led to the loss of a considerable amount of customers' funds. Later this year, the first mining pool was created. A mining pool is an organization of miners that join their computing power, called the hash power, to cooperate in mining blocks on the network. As the the difficulty of creating a new block is dynamically adjusted in proportion to the number of competing miners and the amount of hash power in the network, an increasing number of miners started to participate in mining pools to reduce the variance of their profits and get rewarded in proportion to the percentage of the total hash power they contributed to. In version 0.9.0 the Bitcoin software was renamed Bitcoin Core to distinguish it from the network itself. The protocol has been updated several times over the years to fix various bugs and glitches and overcome potential security risks for the network.

In 2011, Wikileaks and other organizations started accepting bitcoins for donations, mainly due to their censorship-resistant characteristic. In November the 28th 2012, known as the first halving day, the block rewards were halved from ₿50 to ₿25. In 2013, the first Bitcoin ATM was launched in San Diego, California, where you could exchange bitcoins for cash. In the next months, a moderate number of two-way Bitcoin ATMs were installed around the world, for example in Singapore. In the same year the Winklevoss twins filed with the US Securities and Exchange Commission the first Bitcoin ETF proposal. In subsequent years many other Bitcoin ETFs proposals where filed and an increasing number of financial instruments providing exposure to Bitcoin began trading on the exchanges.

Regulatory measures started to be taken by different countries, the US Financial Crimes Enforcement Network classified mining in the United States as Money Service Business, making miners subject to registration and legal obligations. China, amongst other countries, banned cryptocurrency trading and prohibited the use of cryptocurrencies for financial institutions and for buying real goods. Due to the decentralized nature of Bitcoin and the lack of a globally consistent definition of its nature, whether it has to be considered a currency, a property or a financial asset, there is still a lack of a consistent regulatory framework for Bitcoin and other cryptocurrencies as well, but rather and highly fragmented one, which greatly differs by jurisdictions. This also affects taxation in different countries around the world.

In late 2013, Bloomberg made Bitcoin searchable through his platform and, in accordance with the ISO currency code standards, the ticker used was "XBT". In Germany, Bitcoins started to be considered as private money and therefore exempt form taxation if investors held it for a period greater or equal than one year [2].

In 2016, the *SegWit* software upgrade to the Bitcoin source code was proposed and later in 2017 was finally approved and implemented. The aim of this upgrade was to increase the scalability of the network through the optimization of the blocksize, thus lowering transaction fees and increasing the maximum transaction capacity. This upgrade also made the Bitcoin ecosystem compatible with the Lightning Network, an additional protocol built on top of the Bitcoin blockchain, which enabled faster transactions between nodes. Due to a disagreement of some Bitcoin developers and proponents regarding this significant upgrade and on technical details about the size of the blocks, on 1 August 2017 an *hard fork*, was carried out to create the Bitcoin Cash cryptocurrency, which shared all the transaction history of Bitcoin up to that point.

In 2019 the ICE, Intercontinental Exchange owner of the New York Stock Exchange, launched its own cryptocurrencies trading platform called Bakkt, listing Bitcoin options and futures. In 2020 a remarkable number of institutional investors, funds and technology companies started investing in Bitcoin or increasing their stake in the digital asset. During the COVID-19 outbreak and Bitcoin hitting a low of around $4,000, an increasing number of retail and professional investors showed interest in it, primarily seeking the opportunity of making a relevant capital gain in the following months. During the pandemic, Bitcoin kept up with expectations and performed considerably well in comparison with other financial assets and stock indices, registering a substantial recovery from the low of March 2020 and eventually hitting an all time high in December 2020, demonstrating his resilience against time, economic and financial turmoil and negative beliefs of its opponents.

## 1.2 Cryptography

In general, investing in a digital assets or a new technology should not be performed blindly but knowing how it works, where its intrinsic value is derived from, and what are the factors that might influence its financial value.

In order to better understand how Bitcoin really works, the main concepts of cryptography needs to be described. Cryptography has many security and privacy related applications, the main one being sending *encrypted* messages that can be read only by a designated recipient, preventing unwanted third parties from intercepting it. Trough the use of mathematics, the message is encrypted, that is, transformed in an unreadable format called *digest*, and sent to the receiver who will then proceed to *decrypt* it, reverting the encryption through the use of a special key. For example, cryptography is widely used for securing the access to internet web sites, it can be seen from the "https" prefix where "s" stands for secure. The encryption makes the connection between the user and the server, where the web site is hosted, secure from potential malicious snoopers, blocking them from getting their hands on the data flow that is being exchanged, thus ensuring that the connection established is genuine.

In the most basic method of encryption, called *symmetric* encryption, the sender of a message transforms a human-readable *plaintext* according to a single or a set of rules, and send the ciphered message to the recipient who will proceed to decipher the text back, in order to read it, knowing which rule the original message was encrypted with. For example, if a person A wants to send an encrypted message to another person B, he will choose a specific rule to encode the message, like shifting every letter of the message by some determined positions in the alphabet. Let's say the message contains the world 'HELLO', what B will receive is 'JGNNQ' which is the original message with each of its letters shifted by two positions in the alphabet. In order for B to decode the message and be able to know what A really meant, he has to know the rule, called the *key*, established by A beforehand. This simple method of encryption, know as the *Caesar cipher*, has many drawbacks. First of all, it is easy for someone with a lot of patience and a computer to reverse engineer the method used for encryption, for example with a letter frequency analysis, and be able to read the private data contained in the message. Additionally, symmetric encryption is really weak in terms of security because the same key is used for both encrypting and decrypting the message, forcing A to tell B which key is needed to read the message. The need to transfer the key expose both of them to the possibility of an unwanted eavesdropper intercepting whatever type of communication they chose to send the secret key with. Due to its shortcomings, the symmetric encryption is hardly used anymore, the reason being that, nowadays, there is an increasing need of safer layers of security protecting the highly sensitive data underlying internet connections.

In order to overcome these deficiencies, in the *asymmetric* cryptography two different keys are used for encryption and decryption respectively. A user who wants to receive encrypted messages generates the so called *key pair* using a specific scheme of his choice, for example PGP which stands for Pretty Good Privacy, invented by Phil Zimmerman[3]. The key pair is composed by a public key, which the user will broadcast to the public for encrypting messages, and a private key that he needs to keep secret and store somewhere safe, which will be used only by him to decrypt the messages received. These keys, which look like a sequence of random numbers and letters, are mathematically linked together but different from each other, thus eliminating the need of communicating a private key to decrypt messages and avoiding related risks. For example, a user generates a key pair using a software and then can share his public key to everyone who needs to send him a private message containing sensible data. Only in the occasion of receiving an encrpyted message, he will proceed to use his secret key and the senders are confident that only him can read its content.

Bitcoin uses the ECDSA, which stands for *Elliptic Curve Digital Signature Algorithm*, cryptographic scheme to generate the key pairs. As prescribed, the private key is a seventy-eight digits number picked randomly between 0 and $2^{256} - 1$ and the ECDSA computes the public key from it in a way that mathematically links them. A Bitcoin address is derived by the public key and the private key is used to grant access to it, withdraw and send funds to other addresses. In this way, the private key works as a proof of ownership of funds in a specific address and it is used to digitally sign outbound transactions. The user needs to share only his public address to be able to receive payments from others and the ECDSA protocol ensures that it is mathematically unfeasible to derive a private key from a public one. Moreover, the probability of generating a key pair that is already in use is extremely low, given the huge pool of random number the private keys are generated from.

[3]https://www.openpgp.org/

### 1.2.1 Hash Functions

An hash function takes any alphanumeric data string as input, called *preimage*, which is then passed through a set of mathematical algorithms that outputs the so called *hash* or *digest.* There are two main types of hash functions, the basic hash functions, that have a basic linear hashing process, and cryptographic hash functions, which are based on complex and advanced mathematics that yield a more secure encryption process. A basic hash function can be as simple as a position identifier function that outputs the last character of the input string that is passed to, for example:

$$H('simple\,hash\,function') = 'n'$$

Where $H$ is the hash function that wraps the text, and returns the last letter, *'n'* in this case, of the input phrase. Basic hash functions are lacking some characteristics that make them usable for security and safety applications. Indeed, taking as example the $H$ hash function defined above, its main drawback is the non-uniqueness of the digest, namely, the same output could have been obtained with a totally different line of text but ending with an *n.* This hinders its viability for identification purposes, that is why in the Bitcoin network cryptographic hash functions are used instead. The latter satisfy some properties that make them suitable for a cryptocurrency network, where an undeniable proof of the mining process is necessary to link genuine blocks in a chain, the transactions inside the blocks need to be identified by a unique hash and any attempt to compromise the network can be rapidly recognized by the nodes of the network.

Some industry standard crypthographic hash functions, like *MD5* (Message Digest) and *SHA-256* (Secure Hash Algorithm), the one used in the Bitcoin blockchain, are widely employed in modern network security applications because they are deterministic functions, hashing the same input give always the same digest, the hash is computed quickly but it is unfeasible to revert it, that is, retrieve the original message from the hash going backward, and even a small change in the preimage results in a different and unique hash, such that different inputs are linked to different hashes. Another characteristic of the two algorithms mentioned above is that they always output a fixed-length hash, fostering comparability and readability criteria. An example of the *SHA-256* algorithm is:

$$\begin{aligned} SHA256('better\,hash\,function') = {}&'0df3a1b5ba5943318b5876bfbc9c3537886befc \\ &694d1d3f0c1e37ae5db5705d8' \end{aligned}$$

$$\begin{aligned} SHA256('Better\,Hash\,Function') = {}&'eaca307e7d3e732bb30d72818fb09e3408f40b2 \\ &c07f3753b6046ed321758cbd9' \end{aligned}$$

I have used an online tool [4] to generate *SHA-256* hashes for two similar lines of text but

[4] https://emn178.github.io/online-tools/sha256.html

being case sensitive, a slight change in the input makes the output hashes completely different and unique.

### 1.2.2 Digital Signatures

In a decentralized blockchain, due to the absence of a central authority managing users' accounts, there is the need to prove account ownership and a mean to validate transactions between peers in a trusted and secure manner. This is where digital signatures are used. Each user digitally signs outbound transactions using his private key and the recipient is confident that the sender was entitled to move coins from his account and the transaction is authentic. Digital signatures, a subset of the wider domain of electronic signatures, are based on both cryptographic key pairs and hash functions, thus providing a far more secure and reliable way to sign something than any other electronic signature or the classic handwritten signature. The latter can be counterfeited or duplicated for fraudulent uses even without the knowledge of his owner. A digital signature, instead, provides a unique and one-time valid proof for a specific transaction, such that it cannot be reproduced for signing a different transaction. The user that wants to send Bitcoins from his account needs to sign with his private key the hash of the transaction, such that the private key is mathematically linked to it and anyone holding the payer's public key can mathematically verify that this transaction has been generated and authorized by him. Apart for security, one reason why the hash, and not the plain transaction, is signed is to benefit from the fixed length of the *SHA-256* digest, letting the digital signatures to be independent from the length of the underlying transaction or message. The validity of the digital signature, and the integrity of what was signed, can be independently and mathematically be verified by any party, even offline, and both parties can be reassured that the transaction cannot be reverted.

## 1.3 The Bitcoin Network

Bitcoins are the native digital 'coins' of the Bitcoin network, a peer-to-peer system that is freely accessible by anyone through the open-source Bitcoin Core client, whose source code can be found on GitHub[5], or any other compatible software. The network is governed by an hard-coded software protocol, a set of rules that dictates every functional aspect of the system and is what every participant in the network abide by.

### 1.3.1 The Blockchain

Every Bitcoin transaction is recorded on an electronic ledger that, unlike a traditional ledger, is distributed, meaning that it is maintained by a wide number of computer machines disseminated all around the globe, called *nodes*, running the Bitcoin software to access the network. In this way,

[5]https://github.com/bitcoin/bitcoin

the ledger is not controlled by a single entity or a single group of people but is run simultaneously by all the nodes connected to the network that will contribute in managing it.

Thus, according to the protocol, every change in the ownership of Bitcoins is broadcast to every node of the network that will validate each related transaction, flagging them as 'pending', and update its own version of ledger, downloaded through the client. All the pending transactions are grouped together in the so called *mempool* or *memory pool*, waiting to get included in a new *block* of transactions that will be created through the process of *mining* by specialist nodes called 'miners'.

Each new block of transactions is then shared to all the nodes that will proceed to validate it and add it to their own record of the ledger. Each active node constantly listens to new upcoming transactions and repeats the process described above, leading to the formation of new blocks of transactions which will be uniquely linked to the previous ones, thus forming a chain of blocks called *blockchain*, ultimately reaching a network-wide consensus that the chain is genuine and each block contains valid transactions.

The Bitcoin blockchain is deemed decentralized and permissionless because instead of a single entity maintaining it, acting as a gatekeeper that decides who is allowed to participate in the network, it is operated by independent peers, the nodes, and the regulatory burden of performing KYC[6] process, like every financial intermediary has, is replaced by cryptography. Indeed, the creation of an account does not require the permission from the system administrator, the accounts ID are derived from public key cryptography described in section 1.2 and the password to access the accounts and sign transactions are replaced by the user's private key. The account holder can sign transactions even offline and can manage its public-private key pair with a software wallet.

Transaction validation is not performed by the administrator anymore but by nodes who act as bookkeepers of the ledger. Indeed, anyone, anywhere, with an internet connection can become a bookkeeper by downloading the blockchain on their computer. Through a gossip system, every new transaction is propagated between the nodes and the system is resilient as long as there is a consistent number of connected nodes maintaining the same book of records, such that the any attempt of censoring or manipulating transactions would be easily exposed and the malicious node excluded form the network. This is what makes the Bitcoin network censorship resistant as there is not a singe central point of control that maintains records of accounts and balances, chooses which and how to execute transactions. The absence of a hierarchy between nodes makes them important but not essential, as one can go offline and leave at any time without hindering the network functionality.

The reason why each transaction is bundled in a block comes from practicality for bookkeepers when recording and ordering the entries in the ledger. Choosing to work with batches instead of single transactions makes the propagation of new blocks less frequent, thus giving the nodes

[6]Acronym for Know Your Customer, is the process of gathering personal data of clients

more time to acquire them and to agree on their order before new ones are mined, making a network-wide consensus about the correct order of blocks more likely. Once a new block reaches a bookkeeper, it is validated, stored in his copy of the blockchain and propagated to other nodes and all the transactions in it are said to be confirmed with one confirmation. After new blocks are appended to the chain, previous blocks become more and more 'deep' in the network, increasing the confidence that the transactions in it have received enough confirmations to be considered valid.

### 1.3.2 The Mining Process

According to the Bitcoin protocol, new blocks are created each ten minutes on average through a process called 'mining' where each of the specialized nodes compete to find a solution to a cryptographic puzzle before every other miner, earning newly minted bitcoins and the fees attached to the transactions they are including in the block when they provide the proof of their work to the whole network. This scheme is called 'proof-of-work' and helps the network to find consensus about the validity of the entire chain of blocks, firstly proposed by Adam Back with Hashcash [3]. All pending transactions in the mempool require confirmation, so a miner choose to include some of them in a block and leaves a space for an arbitrary number, called the *nonce*, that he will tweak until, once the entire block is hashed, the hash of the block is smaller than a target number defined by the protocol. The hash of the block obtained in this way will be like a fingerprint of the miner and nodes of the network will be able to easily verify that the hash meets the criterion prescribed by the rule, functioning as a proof of the miner's work.

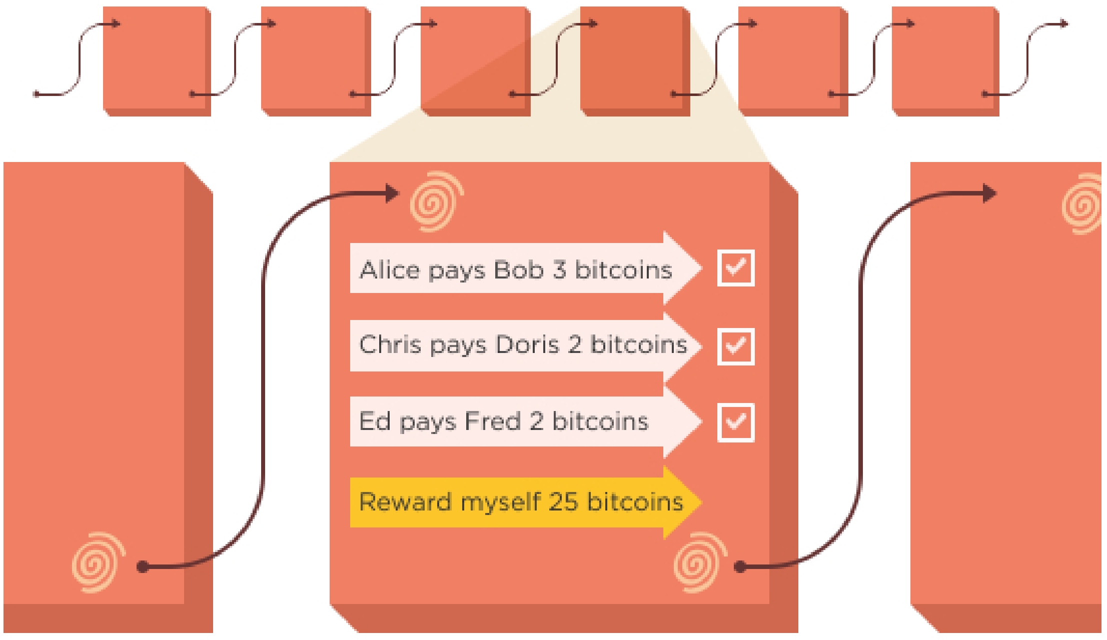
The process of finding the right nonce, called mining, is not straightforward but rather time-consuming and repetitive as the miner is forced to check recursively if the nonce picked is correct by running tedious mathematical algorithms, hash functions. The computational complexity of 'proof-of-work' puzzles requires a lot of hashing power to mine a block[7] thus requiring miners to equip themselves with powerful computers to speed up the process and have a chance to get the block reward. The miner profits from mining new block in two ways. First, he will collect the commission attached to each of the transactions included in a new block that have been specified by the creators of the transactions. This market-based approach makes block-creators prioritize the transactions with the highest commissions therefore making confirmation faster the higher the fee. Secondly, the more substantial reward comes form the coinbase transaction that points to an address specified by the miner and is only transaction in the network that creates new Bitcoins as a reward for mining the block. The maximum amount of new Bitcoins the miners can include in the coinbase is specified in the protocol and is halved every 210,000 blocks, which at 10 minutes per block is approximately every 4 years. The last halving happened on the 11th May 2020 where

[7]As of January 2021, the average hashrate needed to mine a block is about 130 EH/s, one-hundred quintillion hashes per second. Source: btc.com

the reward lowered from ₿12.5 to ₿6.25 per block. Bitcoins have limited supply capped around 21 million units in circulation that will eventually be reached sometime around the year 2140.

As the number of competing miners and their hashing power increases over time, in order to keep the block-creation time equal to ten minutes on average, the protocol dynamically adapts the target for the hash calculations every 2160 block[8] based on the time elapsed from the creation of the last 2160 blocks. The new target is computed from a number called ’difficulty’ such that the network self-balances itself, the faster the previous blocks were mined the higher the difficulty and the lower the target number will be.

To prevent miners from cheating and trying to pre-mine the next blocks, each new block must contain the hash of the previous block as the header identifying it. Thus blocks are not sequentially numerated, which could easily be predicted, but are linked together by previous hashes which cannot be predicted because of their randomness, ensuring that no miner can skip ahead. This principle creates the blockhain, which history is a sequence of uniquely chained and identified blocks and cryptography makes it tamper-evident.



Alice pays Bob 3 bitcoins
Chris pays Doris 2 bitcoins
Ed pays Fred 2 bitcoins
Reward myself 25 bitcoins

Figure 1.1: A Chain of Blocks Linked Together By Hash ‘Fingerprints’ [4]

### 1.3.3 The Consensus Protocol

The Bitcoin protocol contains the ‘longest chain rule’ which determines how the network comes to consensus and dictates what happens when two competing and valid blocks, created by different miners, are broadcast to the nodes at the same time. When there are two valid block at the same

[8]Approximately two weeks at ten minutes per block.

height[9], miners will choose one of them to start the mining process and when done they will add the new block to the chain where in the meantime another new block has been eventually added, thus choosing the longest chain available. The other valid block, called the *orphan block*, is still considered valid but the transactions inside it remain unconfirmed and have to be re-inserted in a new block.

The *longest chain rule* is essential for the resilience of the network against attacks, miscreants and the double-spending problem. Let's say someone sends a payment to another person according to a business contract, the payer broadcasts the payment that will then be added to a valid block and the payee will send the goods as per the contract. Once the buyer has received the goods, in theory, he could start a longer chain where he replaced the transaction containing the payment to the counterparty with a payment to himself, thus deceiving the payee as the network will accept the longest chain by protocol and the block containing the fraudulent transaction will be deemed valid. Therefore it is best to wait the block containing a transaction to be enough *deep* in the chain such that it becomes more and more confirmed when enough blocks are added after it, and then send the goods. In order to subvert the network and change or reverse transactions at will, someone should have enough hashing power to re-mine the blocks he wants to tamper, for example to be able to double spend like in the previous example, but he must also keep up with the the honest chain and eventually surpass it to make all the nodes in the network choose his. This requires an exceptional amount of hashing power and as described in the white paper, unless the malicious miner has more than the 50% of the total hashing power of the entire network, his probability to revert older blocks and catching up the other chain, beating the other miners, drops exponentially as the difference in terms of number of blocks between the chains increases. The so called 51% attack happens when the dishonest miners have the potential to undermine the entire blockchain and the honest nodes cannot keep up in terms of computer power, that is why the more hashing power in the network, the more resistant to this kind of attacks it will be. Although 51% attacks are possible, nowadays the popularity of Bitcoin has made it more robust, as a matter of fact, given that the hash power to mine Bitcoin is now exceptionally high, the computational power required to overtake the blockchain would be enormous and supposing that a miner, or a group of them, has enough power at their disposal to perform an attack, they would be discouraged to do so as the mining equipment they already have is worth millions, the electricity costs they bear should be already astronomical, if their actions would undermine the confidence of network participants and of the Bitcoin market, causing the value of coins to plunge, they would be left with a lump of worthless coins in their wallets, thus making the attack counterproductive.

A malicious miner could deliberately choose which transactions to add or exclude from his block, effectively censoring data, or try to double spend bitcoins but they would be eventually excluded from the network if the other participants reach consensus that he his behaving against

[9]The height term is used to identify the position of a block in the chain.

the authenticity of the network itself and other miners can agree to not mine other new blocks on top of his. Miners, however, cannot steal coins from other users as long as they don't have access to their private key to sign transactions and cannot create Bitcoins out of thin air because that would mean go against the protocol and have their block refused by nodes. Malicious bookkeepers can try to withhold transactions and blocks or share a fraudulent version of the blockchain to other bookkeepers but any discrepancy form the copies of the majority of other nodes can swiftly be exposed, getting them excluded from the network.

### 1.3.4 How Transactions Work

Differently from other ledgers, the blockchain does not store accounts balances but transactions between every address registered in the network so one has to infer from all the inbound and outbound transactions how many Bitcoins are available at a specific address. Give that only the coinbase is able to mint new bitcoins, the ordinary transactions move Bitcoin between the addresses, for example, let's say A have 5 Bitcoins available from previous transactions and wants to send 2 Bitcoins to B, the transaction he creates to pay B will move all A's Bitcoins, that is 5, sending ₿2 to B's address and the remaining back to A's account. If later A wants to spend the leftover Bitcoins he will create a new transaction that will take as input the output of the previous transaction, namely ₿3. More specifically, to perform a payment, the user needs to specify which coins, called UTXOs[10], to move by referring to the hash of their related transaction. By this way, identically to blocks, every transaction is linked to the hashes of prior transactions, creating a public and transparent chain that can easily be inspected to trace every movement of a specific lump of bitcoins, including every account it went through, all the way back to the coinbase source.

Afterwards, the transaction is created and signed, giving it an hash ID, and sent directly to the peer-to-peer network of bookkeepers that will proceed to validate and broadcast it to all nodes. Unlike client-server systems with a central third party, the peers of the bitcoin blockchain will act as intermediaries but with no seniority, data is shared between them through the internet connection and even if some nodes disconnect from the network, accidentally or deliberately, the network will continue to work as data is replicated and updated simultaneously by all remaining nodes.

Unlike one might think, Bitcoin wallets do not store coins but a copy of the private key of one or more addresses. There are different types of wallets with distinct features fulfilling many purposes and users needs. Software wallets can be used to create a bitcoin account, safely store the private key, share the address to others, like through a QR code, sign payments and check the balance of an account. To perform these tasks, the wallet needs to connect to the blockchain and does this in two ways, either acting as a full node by downloading the full blockchian, which is worth more than two-hundred gigabytes, or by storing a lighter version of the blockchain and

[10]Unspent Transactions Output.

connecting via server with a full node, relegating all the heavy tasks to it. While a full node needs to maintain an ongoing internet connection in order to keep the blockchain up to date, lightweight nodes query the server hosting a full node when performing tasks on behalf of the user, making it better suited for less powerful machines like mobile phones. When making a transaction, the wallet gathers all the needed information and then sends it to the network. Some software wallets offer security functionalities like private key backup and password encryption and some of them are connected to either cloud services providers or to exchanges for quick transactions between different cryptocurrencies. Some wallets can perform private key *sharding*, namely splitting the the private key in multiple pieces and requiring a certain amount of *shards* for the key to be used. There are wallets that make possible to create *multi-signature* addresses requiring multiple private keys to approve a transaction. Hardware wallets instead are much like OTP tokens provided by banks, small handheld devices that are not connected to the internet and safely store the user's private key. They are equipped with a chip that perform basic pre-programmed actions like signing an incoming transaction with a click of a button. Of course private keys can be kept on a machine not connected to the internet, on an unwired hard disk or even written on a piece of paper or multiple bits of paper each stored in locked deposits in different banks, these methods are called *cold* storage. Hot wallets instead, are automated wallets that require little to no human intervention to perform tasks like creating and signing transactions and are often used by exchanges that need a quick method to meet their clients' sudden withdrawal requests. In this case, the private keys are stored on online machines thus making them more exposed to hackers and cyber attacks, making them less safe, which is the reason why exchanges keep only a small part of their total coin holdings in the related addresses.

## 1.4 Criticism

Since its implementation in 2009, Bitcoin has been heavily criticized for multiple reasons but have also received several appraisals and positive opinions, thus gaining more and more supporters and interest from investors.

### 1.4.1 Ideology Behind Bitcoin

Firstly, the ideology that shines through the words of Nakamoto, "The root problem with conventional currencies is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust." [5], has been criticised by many economists to be techno-anarchist and, at least initially, that Bitcoin has attracted mostly populism and libertarianism advocates for its potential to detach money from the governments control and not for its investment or real value. In line with an article on the *New York Times* and the argument by Bitcoin early supporter Roger Ver, in the early stages, the first cryptocurrency gained the most attention by the people who embraced the idea of subverting the modern financial system and jailbreak from the stranglehold of central banks' monetary policies. From the *The Declaration of Bitcoin's Independence*, the philosophy of the so called crypto-anarchists stems up, sinking its roots in the idea of supporting a disruptive technology that can withhold the control of the wealth in the system from malicious interference of banks and states, endorsing humanitarian values.

### 1.4.2 Regulation and Legal Concerns

Throughout the years, the US Commodity Futures Trading Commission, the SEC and the European Banking Authority, amongst other institutions, issued several warnings, alerting investors of the risks associated to trading and investing in cryptocurrencies due to their extremely speculative nature. The main financial risk comes from the high volatile environment of cryptocurrencies, including Bitcoin, and when analyzing the risks attached to them, one cannot refrain from considering that due to their decentralized nature there is often neither an authority nor an institution backing them, and in the occurrence of a security breach of their network or of the exchange's system, a bug in the source code or another uncontrollable event, they can sink and go bust, without anyone bailing them out. Ultimately, retail investors have often been exposed to frauds on the web, for example solicited on social media sites, Ponzi and pyramid schemes, deceived by false promises of earning profits and/or cryptocurrencies. Some cases of price manipulation, such as spoofing and wash trades, have been investigated and exposed by the competent authorities.

Bitcoin, along other cryptocurrencies, has often been associated with its use for illegal purposes, and its censorship-resistant feature has been accused of facilitating money laundering and the use of it by criminal organizations and terrorists. If on one hand Bitcoin has been the main virtual currency to be accepted by the shady merchants of the illicit *deep-web* marketplace, it has to be considered that by now, the use of Bitcoin for illegal activities is just a small percentage of its total market value and its increasing adoption functions as a deterrent for bad actors who seek to use it for unlawful acts. As a matter of fact, even though the creation of a Bitcoin account for sending and receiving payments is completely anonymous, the Bitcoin blockchain is completely transparent and every single transaction can be traced back from its inception and all the capital flows can be tracked by accessing many websites like *blockchain.com*, or if one has the need of getting the complete chain examined, the whole Bitcoin blockchain can be downloaded on any computer. Additionally, fiat currencies still remain the most used for illegal activities in defiance of the law as cash is really anonymous and untraceable.

### 1.4.3 Resources Consumption and Centralization Concerns

Bitcoin has been targeted by many climate change activists, associations and scientific organizations for its environmental impact and power consumption related to the mining process. As a matter of fact, the significant amount of computer processing power required for mining Bitcoin, which as prescribed by its protocol dynamically increases as new miners enter the network, raised concerns about the electricity demand of the equipment designated for this type of process. As reported by the *BBC* [6], by the end of 2019 the Bitcoin electricity consumption in gigawatts was comparable to that of the entire country of Switzerland, and estimated to be the 0,2% of the global electricity consumption. Conversely, a study performed by the American journal *Politico* [7] claimed that even though the Bitcoin electricity demand is high, its power consumption is considerably lower than the levels of the whole global banking sector, and when approximating the future maximum consumption level the Bitcoin network might require according to its technology, this projection would still be under 2% of the global power consumption.

Thus, due to the power hungry mining rigs, the vast majority of mining farms, huge warehouses where several high-performance computers are connected together to generate the most hashing power, are often geographically located in areas were the cost of electricity is lower than the average and the climate conditions are more favorable. Most miners opted to locate their farms in Iceland for example, in order to exploit the cold temperatures and cut the cost of computer-cooling equipment. Moreover, as of 2020, a prevalent portion of mining farms and pools are based in China, the main reason being that miners are able to exploit electricity subsidies provided by the Chinese government, resulting in more than 50% of the global Bitcoin hashrate concentrated in the PRC[11]. This brings up the concern of a centralization of the computing power needed to generate new Bitcoins, going in the complete opposite direction of the decentralization on which Bitcoin is based on and the ideology at long avowed by its proponents. Considering that the most widespread and preeminent way of acquiring Bitcoins, and other digital currencies, is trough exchanges, centralized by definition, a perfect decentralization is farther than one might hope.

During the years, the exponential increase of the hash power needed to mine Bitcoin fostered the proliferation of mining pools, some of which, like *BTC.com* and *AntPool*, owned by the same Chinese company Bitmain, are estimated to control a prevalent portion of the total Bitcoin hash power. Moreover, some of these pools started selling ASIC[12] powered mining hardware, raising concerns about the centralization of miners, hash power and mining equipment manufacturing and distribution. This exposes the network to the risk that if the biggest mining pools join their forces, resulting in more than 50% of the total hashing power, they could undermine the network by double-spending and re-writing blocks. Even if they might get discouraged because doing so would wipe the confidence in Bitcoin, making their holding practically worthless, one might argue that they could open a short position and profit from the price plunging.

Additionally, the vast majority of nodes are running on the same software, the Bitcoin core,

[11]Source: cbeci.org

[12]ASIC stands for Application Specific Integrated Chips. These chips are specifically designed and optimized for SHA-256 hashing, giving them a competitive advantage over general-purpose CPUs and GPUs not only in terms of hash power but in energy efficiency and power consumption.

which is maintained by a single organization and developed by a small amount of developers, in response to this concern it must be noted that the Bitcoin software is open-source an anyone can contribute to its development.

### 1.4.4 Noteworthy Criticisms and Future Outlook

At last, many economists and Nobel laureates, like Paul Krugman and Robert Shiller, labeled Bitcoin as a bubble that will explode eventually. Renowned professional investors, like George Soros and Warren Buffet, referred to Bitcoin as poised to die and disappear. In response to these heavy criticisms, the community around Bitcoin even created a web site that keeps counting each time Bitcoin has been declared dead, attaching the relative citation. Even if Bitcoin has its own drawbacks and vulnerabilities, like every new technology that presents itself as having a revolutionary potential, it needs to be well-known and assimilated by the vast majority of actors in the economy for its value to be really appreciated. The blockchain concept and its capabilities, as well as the cryptography based security measures, should not be overlooked, and its crucial to evaluate their usefulness and applicability in every economic sector. The decentralization of a system contributes to avoid the risks associated with having a single point of failure, like most of the modern electronic and software systems, where a negative feedback loop could propagate from, affecting every unit and its participants.

The shift towards a complete digitalization of money seems the next step to take in our modern society, electronic payments are now a standard, and an increasing number of Fintech companies are participating in the financial sector, urging their traditional competitors to look for technological innovations to invest in. The adoption of the blockchain technology by many leading financial institutions, like for example JP Morgan Chase with Quorum, is representative of how much this technology is set to transform the economy and affect financial services in the near future. Every company need a fast and reliable database to store important data about their business or need a business-to-business communication system that ensures a tamper-resistant mean of sharing vital information, like contracts, reports or even digital assets. That is why firms are consistently starting to explore this new technology and develop private blockchains, although relaxing some of the concepts and requirements of public blockchains, to overcome the shortcomings of traditional database systems that have not received many radically innovative upgrades during their life cycle. For example, they can benefit from the automation of specific processes through pre-programmed smart-contracts based on the Ethereum platform, or increase the quality and security of data flows getting rid of the costs and risks, like leakage or mishandling, related to a central repository or a third party whose failure would have a significant impact on the entire business process.

## 1.5 Bitcoin Economics

### 1.5.1 Definitions and Use Cases

When referring to Bitcoin, there is no universal definition for it, some may call it a currency, given that it is a cryptocurrency, some say that it is a store of value, others, especially in the financial sector, purport that digital speculative asset might be the most suitable definition for it. As a matter of fact, when comparing it to the academic definition of money, Bitcoin is lacking some necessary attributes required to be considered as a proper currency.

First, it needs to fulfill the medium of exchange function, it has to be widely accepted as a payment method in a commerce environment, namely, used to buy tangible or intangible goods, pay for services, or extinguish debt and financial obligations. Although its popularity, Bitcoin's inherent volatility drags most merchants away from accepting it as a directed payment method like traditional currencies because that would mean exposing themselves to the risks related to frequent exchange rate fluctuations. This does not mean that there aren't exceptions and even if during Bitcoin's early years, where its technology had to be explored and appreciated first, there were an inconsistent number of companies willing to accept Bitcoin in exchange for their goods and services, nowadays there are some notorious names in the list of retailers allowing payments with Bitcoin, notably, Microsoft that halted its acceptance but later resumed it to buy digital goods on its online marketplace, American telecommunication company AT&T and well know fast-food restaurant chains and even online computer hardware and accessories vendor NewEgg.com. This list is enlarging as time goes by, mainly due to agreements between merchants and cryptocurencies exchanges offering competitive commission fees schedules for in-store and real-time crypto-fiat currency exchange with respect to other POS and electronic payments providers, like Visa and MasterCard.

Secondly, following the definition, money has to fulfill the store of value function, meaning that individuals in the economy should be confident in holding their wealth denominated in a currency such that the same amount of money that would take them to buy a specific basket of goods remains the same as time goes by. In reality, even for fiat currencies this does not hold well due to inflation, induced by central banks' monetary policies, that erodes the purchasing power of a specific currency. This drives investors to look for a way to hedge their wealth against higher expected levels of inflation inducing them to seek for opportunities on the financial markets, like stocks, ETFs and commodities. Analyzing the historical evolution of inflation in the U.S. represented by CPI index[13], it is evident that the dollar, like any other government currency, failed to ensure an adequate hedge against inflation over the last hundred years, and even more dangerous to the stability of a state-issued currency would be the phenomenon of hyperinflation which would crumble its value.

[13]https://fred.stlouisfed.org/series/CUUR0000SA0R

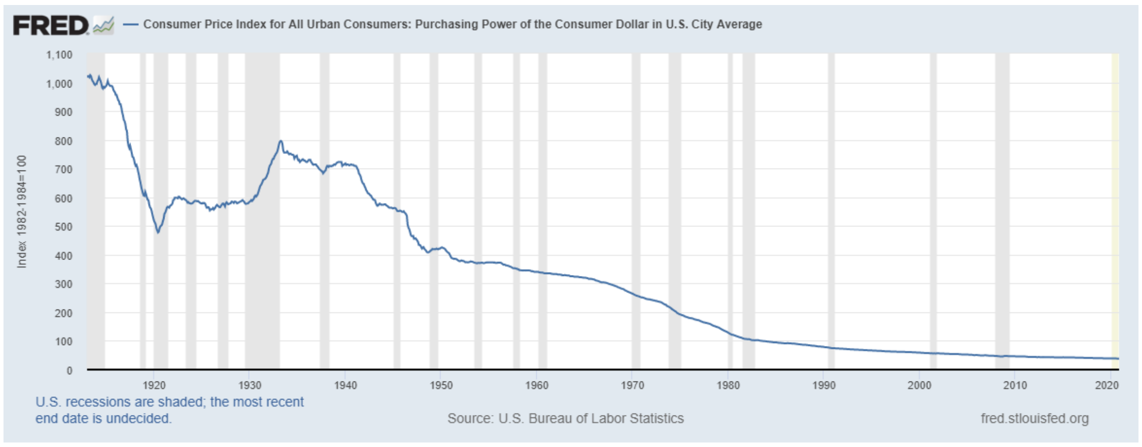


Figure 1.2: Consumer Price Index: Purchasing Power of the Consumer Dollar in U.S. City Average.

Bitcoin has often been labelled as the 21st century gold, digital gold or gold 2.0, and under certain aspects this comparison may probably be accurate but has to be supported by an accurate analysis of the assets' respective intrinsic characteristics. Both assets have limited supply, gold is by nature a limited resource and Bitcoins in circulation will be at maximum 21 million, today's circulating supply is approximately 18.5 million[14] and geometrically decaying with time as designed by its protocol. What makes gold good at fulfilling this particular function is the perceived value the economy has of it, apart from industrial applications, like in the electronic circuits sector, and the jewellery sector, which is itself driven by the value of gold more than the other way around, it has even been the chosen *standard* for backing fiat currencies in the past, so is considered by people as a secure way to store their wealth outside the traditional banking system and attracts many speculators due to its inherent value. On the other hand, in response to the criticisms that Bitcoin has no intrinsic value and there is no authority backing it, it should be noted that are exactly its innovative and disruptive characteristics, like decentralization, lower intermediaries, and the clever use of cryptography, that propagate confidence and trust between its users, bolstering it effective utility. In an historical period where the use of cash is decreasing and digital payments are sometimes preferred either for their convenience or when they are the only way to pay for exclusively digital goods or services on web-sites, it is only a matter of time before more actors in the economy would want to benefit from Bitcoin, and cryptocurrencies in general, capabilities. Additionally, the pre-programmed supply limit has deflationary effects, thus helping to maintain the value of Bitcoin high in the future due to its inevitable scarcity as long as demand for it stay constant or increases, miners will stop getting block rewards so the market for fees should become more competitive. But, citing Robert Sams, "the downside of a known, predictable, and completely inelastic supply unrelated to a fluctuating demand results in perpetual price volatility" [8].

The last property for money to be considered as such is the unit of account, meaning that agents in the economy should reliably use a currency as a benchmark for comparing different assets or

[14]Source: https://coinmarketcap.com/

evaluating their total holdings. As a unit of account, people tend to estimate their total wealth in terms of a specific currency thus needing a fairly stable one to peg their possessions to. The high volatility of Bitcoin hinders its viability as a conventional unit of account and is the main reason why most merchants are restraining form pricing their goods or services in Bitcoin. On the contrary, Bitcoin functions as vehicle currency for a manifold of *altcoins*, ultimately satisfying the unit of account properties for the cryptocurrencies market. Altcoins is the definition for the enormous amount of cryptocurrencies that surged after their precursor, either by forking Bitcoin codebase or by *chainsplits*[15] like Litecoin and Bitcoin Cash respectively, and especially the ones with a low trading volume and market capitalization are quoted and traded only in terms of Bitcoins similarly to USD in respect to other fiat currencies. The thicker volume of USD markets ensure a lower bid-ask spread and is often more convenient to take an extra step trough USD when exchanging two currencies than a direct exchange between them. As the Bitcoin-USD market is the thickest in terms of volume, it is often cheaper, or the only way, to exchange USD for BTC and BTC for altcoins, making Bitcoin an effective unit of account for most altcoin markets.

Some merchants are adding an additional payment method amongst the others to accommodate users needs and to capitalize on the wave of Bitcoin's popularity but are still requiring an additional service of a third party intermediary to perform seamless crypto-fiat exchanges as the majority of them prefers to have their money in the currency of the jurisdiction where they operate. One function where Bitcoin might do well and be highly competitive is in international remittances. As a substantial portion of this market is taken by two established companies, Western Union and MoneyGram, that charge quasi-monopolistic taxes for sending money abroad, there is where Bitcoin might shine. Indeed, someone who need to send a sum of capital to another person overseas could use Bitcoin as vehicle for other fiat currency and even without a BTC address one could access a simple Bitcoin ATM and by showing the QR code obtained therefrom to a local "reBITtance" office, exchange bitcoin for cash right-away or instruct the service provider to exchange BTC for fiat and send fiat to a the recipient's bank account (Ferraz 2014 [9], Buenaventura 2014 [10]).

### 1.5.2 Is Bitcoin a Bubble?

Bitcoin and other cryptocurrencies are not redeemable at the state bank for fiat currency or for any kind of commodity, still the pre-programmed cap of their supply prevents the risks related to an overissue that would affect their market price. As seen before, the fixed supply leaves the

[15]The fork of a codebase happens when the original source code of a cryptocurrency is customized by enriching its features, adding some improvements, tweaking its original parameters or radically changing its protocol like block generation and consensus procedures, departing from the proof-of-work, thus creating a blockchain from scratch which will support a new cryptocurrency that will be traded with a new ticker symbol. A fork of a live blockchain, in jargon a chainsplit, instead stems from a disagreement between some members of the community of a cryptocurrency, often about the modus operandi of the network, leading them to create a new cryptocurrency governed by a different protocol that will share a common history with the old cryptocurrency up to a certain block.

demand to move freely and drive the price, thus resulting in an highly volatile environment which for some cryptocurrencies, whose demand was not supported by any inherent usefulness but just by speculation, was lethal and their prices plummeted to zero wiping their entire market cap. This raised concerns about the instability and riskiness of most altcoins markets that were inevitably labelled as speculative bubble as their value did not have any floor under their equilibrium market price supported by a sovereign authority or by utility in other contexts like commodities. According to the monetary supply theory, agents in the economy might want to hold an irredeemable currency if it is a mean of exchange or because they have positive expectations about their future value, but if there is no price-independent characteristic backing it, its demand is purely speculative and expectations driven, thus falling in the definition of a bubble. But this does not mean that all cryptocurrencies are bubbles as the ones with the highest market cap have many fundamental values sustaining the price from piercing a certain threshold and explaining their positive market value. If one looks from the prospective of a tech-enthusiast, a crypto-developer, a member of the cypherpunk community or even someone whose endorsing the ideological values behind a certain cryptocurrency, holding Bitcoin, or Ethereum for example, could be like holding gold or a rare numbered piece of art, to the extent that often there is a psychological support level for their market price that when reached would steadily trigger a sustained demand and bring the price back up. The other price-independent features that would sustain the price of Bitcoin are the confidence in the use of cryptography and its technology in general, adding more value to the mere expectation of Bitcoin becoming a widely accepted medium of exchange, although the latter remains beneficial for its market price as it would increase the liquidity premium over its fundamental value.

Velde (2013) [11] curbed the enthusiasm towards Bitcoin along with the prospect for it to become widely accepted and compete with the U.S. dollar, criticizing the fact that its software is maintained by just a small set of programmers by saying:

> "*Although some of the enthusiasm for bitcoin is driven by distrust of state-issued currency, it is hard to imagine a world where the main currency is based on an extremely complex code understood by only a few, and controlled by even fewer, without accountability, arbitration, or recourse.*"

In response to that, Lawrence H. White (2015) [12] pointed out that if one replaces the word *code* with *bureaucratic agency* this statement becomes a fair description of the modern Federal Reserve controlled currency system thus leading him to purport that the drawbacks and shortcomings of Bitcoin could be offset by the benefits of a public and cryptography secured public ledger when compared to the byzantine central bank.

Bitcoin can be traded on exchanges with the ticker BTC and there are BTC-fiat currency pairs available for major traditional currencies, moreover, beyond established and conventional brokers, there is a cluster of crypto-to-crypto exchanges where BTC is traded in pair with other cryptocurrencies only, and in order to get a sense of how much BTC is worth in terms of fiat currencies there is a large number of so-called *stablecoins* in circulation, digital tokens that are programmed to be pegged to a specific fiat currency, like EUR or USD, but not without flaws, such as technical difficulties to peg a digital currency to a real one, and risks about how these coins are managed by their issuer. Indeed, some of these stablecoins are minted by private companies, like in the case of the popular USDT, operating in a total legal grey area and often do not undergo the prudential audits envisioned for traditional financial companies, thus the mishandling of their reserves can expose crypto-markets to serious stability and systemic risks and some of these companies have already been accused by market participants of allegedly flooding the price of cryptocurrencies and not being transparent enough about the mechanism and the rate at which their stablecoin is backed with real money.

### 1.5.3 Price Analysis

I downloaded the historical data of BTC/USD from *Coindesk.com* for the period going from 31 December 2020 all the way back to 1 October 2013, which was the earliest date available. Then I plotted the time series with Python library *Matplotlib* to obtain a subplot of the Bitcoin price in dollars in order to analyze its historical behaviour:

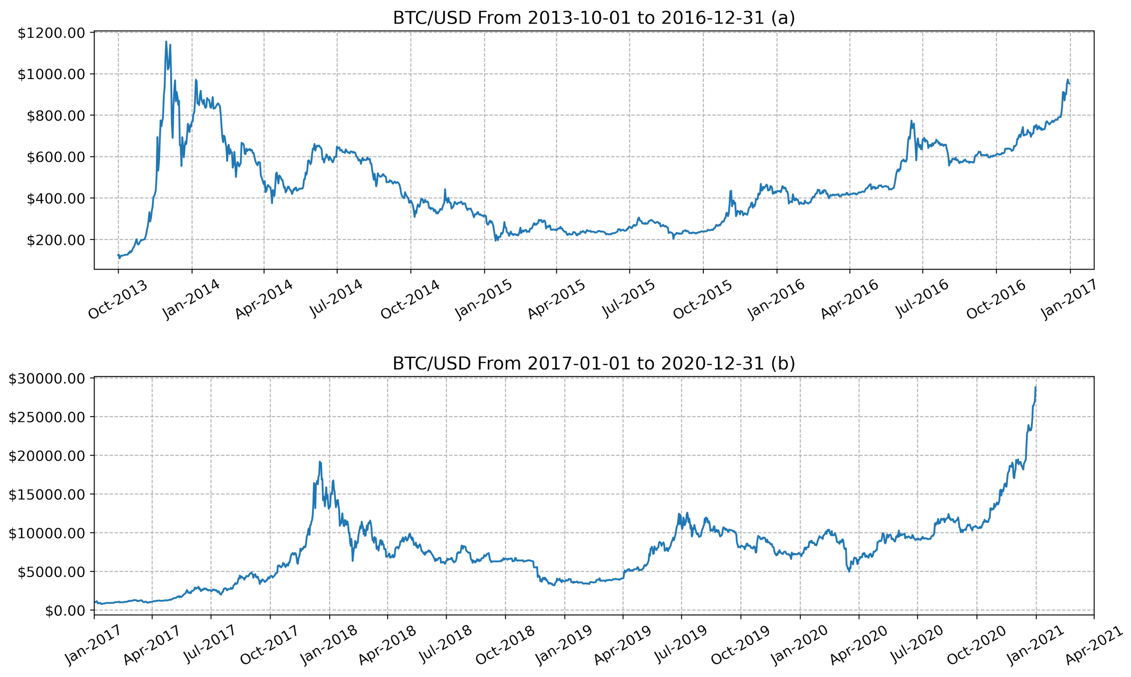


Figure 1.3: Bitcoin Historical Price Evolution

From Fig. 1.3 (a) it can be seen that after reaching an all-time-high of around 1,200 USD in November 2013, Bitcoin price plunged to $600 in the first months of 2014 when Mt Gox exchange, which accounted for a significant portion of total volume, started its tormented path through scandals and hacks and ultimately going bankrupt. Moreover, in the first quart of 2014 the Chinese government announced that it would have banned cryptocurrencies trading and commercial uses such that the price crumbled, getting back to $200. Reaching another peak by the end of 2017, Fig. 1.3(b), Bitcoin was entering what has later been labeled as the the great crypto crash, a substantial sell-off that can be linked to various factor happened in that year. The largest cryptocurrencies exchange in Japan, Coincheck, disclosed an loss of over 500 million US dollars after being the target of an hack, spreading fear amongst investors that their money were at risk and started withdrawing from the network, and the tough criticisms by prominent individuals in the financial sector, like Warren Buffett and hedge fund managers, worsened the confidence people had in cryptocurrencies in general. Additionally, in that year Bitcoin Cash was created from a Bitcoin chainsplit and the community that was agreeing to the hard fork was emphasizing Bitcoin shortcomings, in part to lure the attention and capital in favour of the new cryptocurrency, making the price of BTC to slide under the $5,000 price point in the final month of 2018. After regaining some momentum in the first two quarters of 2019, the priced declined back to its level in Q4 when Russia started enacting regulations towards smart-contracts and cryptocurrency tokens. In 2020, Bitcoin, together the whole financial market, registered a substantial loss of more than 50% in March 2021 following the beginning of the COVID-19 pandemic that heavily affected most economic sectors. Despite the economic downturn, Bitcoin held up and started from the low of $4,000 to achieve higher than average returns over the last half of the year and ultimately sky-rocketing to a new all time high and contributing to set the $1 trillion total market capitalization milestone of the cryptocurrencies market.

The are two main factor that could help to explain the performance of Bitcoin from April 2020 until the end of the year. First of all, in a bleeding economy where the coronavirus disease forced the population to stay home and many companies to shut down and eventually declare bankruptcy, BTC steadily rose in value along with the tech sector because investors closed or resized their position in losing sectors and moved their funds where they saw an opportunity to buy undervalued assets. Moreover, most countries' central banks and authorities started to implement policy measures providing financial aid to support and stir up the economy, an example can be the U.S. Cares Act by which the American government envisaged a stimulus package to contain the effect of the economic fallout such as helping small-medium sized business, loosening taxes and interests on loans and ultimately providing a one-time check payment to individuals and families. In a critic environment, the shift to a more expansionary monetary policy leads to a reduction of interest rates and investors might build an expectation about an higher inflation in subsequent months and years such that they start investing in assets they deem to be an efficient store of value to protect their capital from inflationary erosion. Moreover, in the last two quarters of 2020, many companies and hedge funds, like Microstrategy and MassMutual, announced that they would take or increase their position in Bitcoin, and in the October 2020 PayPal opened its digital payments platform to cryptocurrencies, allowing its subscribers to buy and sell crypto directly form their PayPal dedicated crypto wallet, making transactions in Bitcoin easier for users. The adoption by large institutional investors drove an unprecedented bull market that lasted through the last months of 2020 and the number of Bitcoin mined, transferred and the number of new accounts created hit record values[16].

Remarkably, Bitcoin market capitalization is approximately 700 billion dollars and accounts for 70% of the total cryptocurrencies market cap[17], confirming Bitcoin dominance among this market. But when comparing these values with the market cap of other well established assets like gold and tech giants like Apple (figure 1.4), Bitcoin market cap is conspicuously lower, signaling that despite being around for more than ten years, its market is still in his early days and mass adoption is far from being reached.

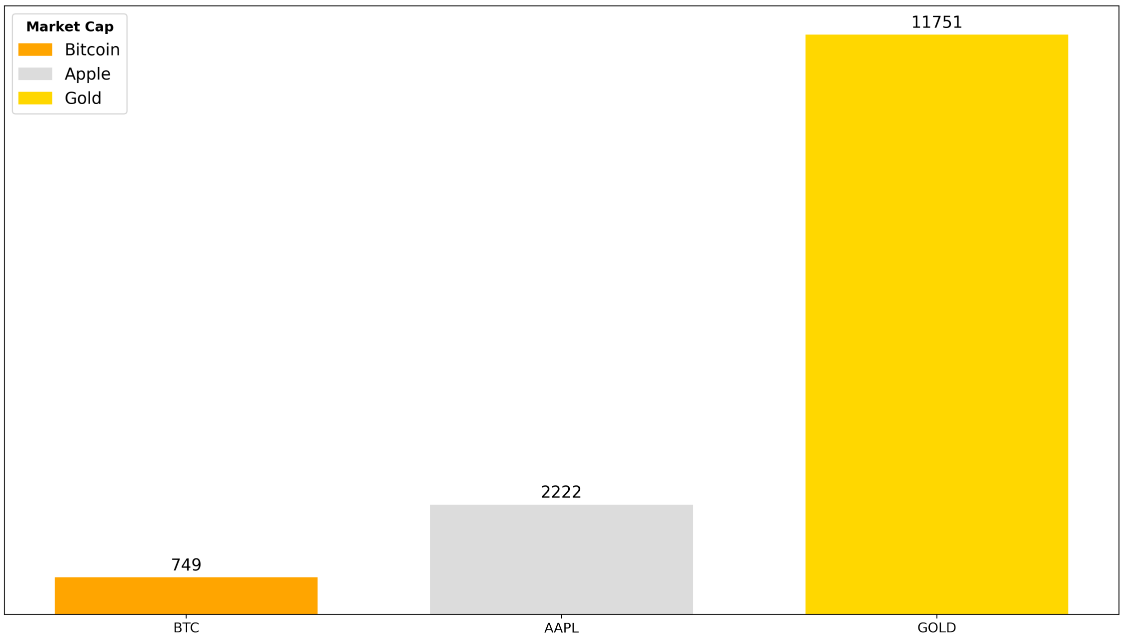


Figure 1.4: Market Capitalization of Bitcoin, Apple and Gold in billion U.S. dollars

The small market cap makes investing in Bitcoin more riskier and this accounts for its higher returns when compared to traditional assets. The higher volatility does not help with wide acceptance and Bitcoin needs to prove itself for a longer time frame before it can be considered like gold which have been recognised as a store of value for over 300 years. In figure 1.5 I plotted the compounded gross returns of daily prices of Bitcoin, priced in US dollars, and gold futures, ticker GC=F traded on Comex exchange,[18] to grasp the performance of both assets in the time span between 2017 and 2020 and I added the mean of their respective compounded returns series to get an idea of their long term volatility. Price fluctuations around a fixed level of Bitcoin are

[16]Source: chainalysis.com & wsj.com
[17]Source: coinmarketcap.com/
[18]Source of the data: finance.yahoo.com/

noticeably higher and the latter has been characterized by a larger price excursion in some days when the percentage change was higher than average. I decided to leave the gross return to better understand how much an investment made in the first day of 2017 would have been worth at the end of December 2020, for example an investment of $1,000 dollars on the 1st of January, 2017 would have been valued approximately $30,000 dollars on 31 December 2020, that is around a 2,900% return in two years, not bad for a digital asset.

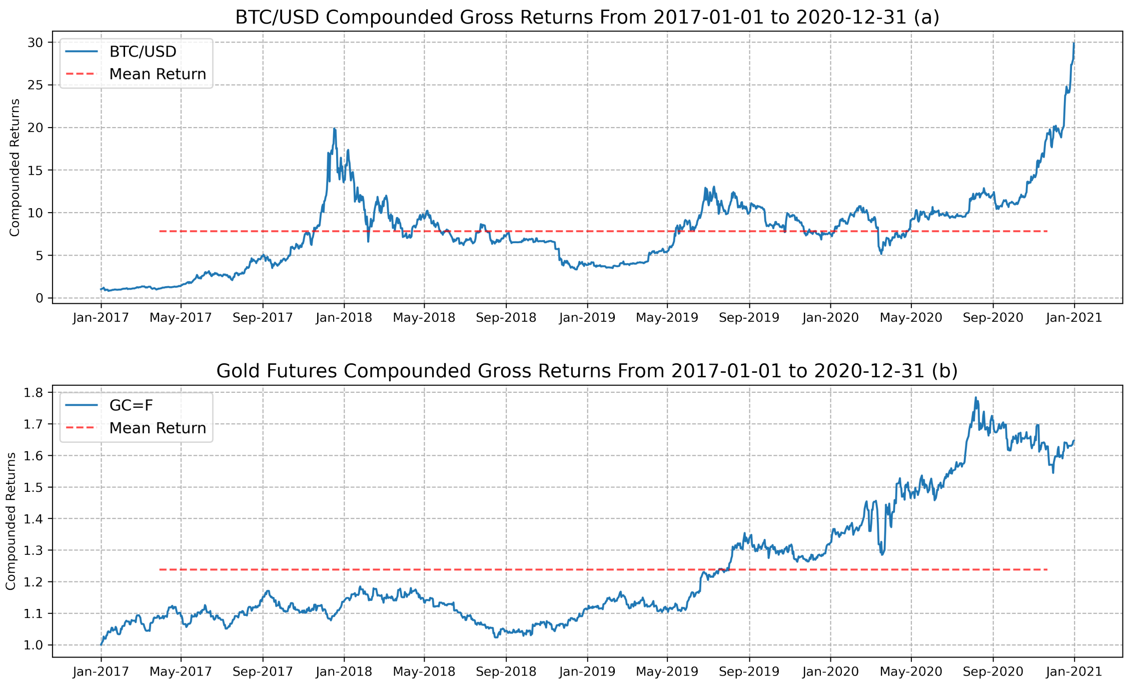


Figure 1.5: Bitcoin and Gold Return Comparison

Finally, a factor that contributes to the high volatility and the speculative nature of Bitcoin is the lack of a widely accepted and used pricing model and, differently from stocks where investors can perform duly fundamental analysis to get valuable insights to build their decisions on, there is neither a financial statement or report related to BTC nor an academy acknowledged method of valuation, but still, information on its blockchain and data about the network is publicly available on a variety of trusted web sites and could provide a good understanding of the price evolution. In order to get a general overview of the performance of different markets, I plotted in Fig. 1.6 the compounded percentage returns for the entire year 2020 of different assets and indexes, namely BTC/USD, the S&P 500 US index, the FTSE MIB Milan stock exchange index, silver (ticker GC=F) and crude oil (ticker CL=F) futures traded on COMEX and NYMEX respectively, and EUR/USD currency pair. As it can be seen form the chart, the Bitcoin has managed to gain massive returns in comparison to more traditional assets and its price evolution is quite volatile, this is the main reason why in the next chapter I have chosen to employ an econometric approach to model the time varying conditional volatility of BTC along with its conditional mean, trying to find a model that provides a suitable fit for the time series of Bitcoin.

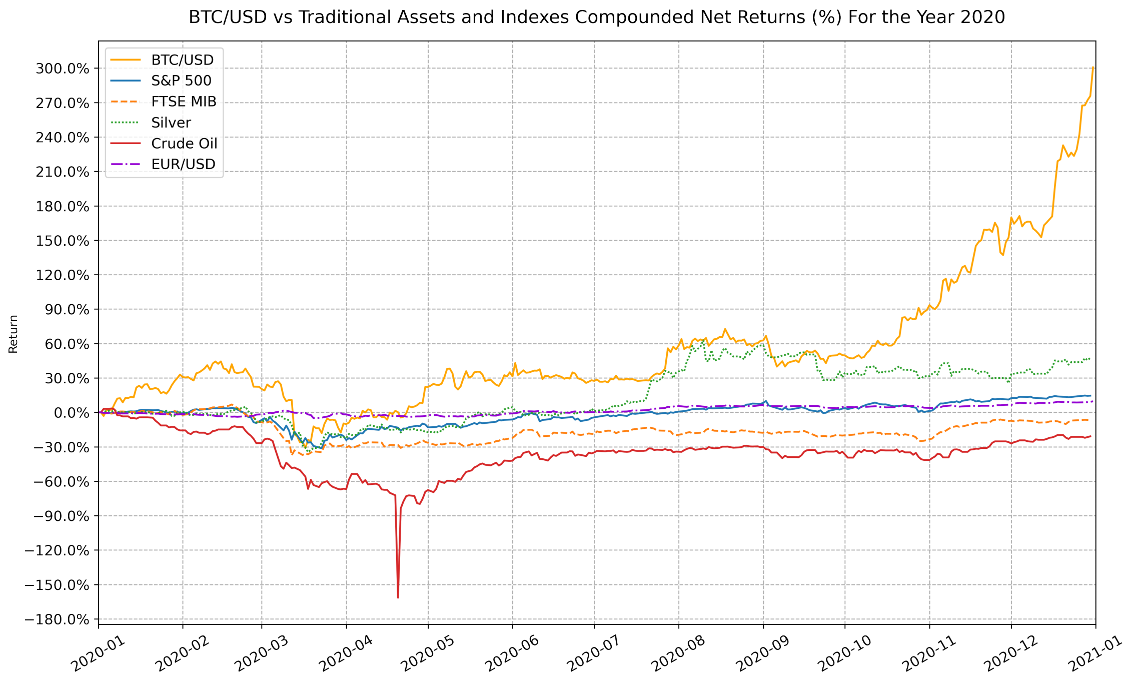


Figure 1.6: Comparison Between Returns

# Chapter 2

# Time Series Analysis of Bitcoin

In this chapter I will apply ARIMA and GARCH econometric models to perform a univariate time series analysis and prediction of Bitcoin price series. I have downloaded the 4 hours close prices through Python, calling the API[1] of Binance exchange with an open-source wrapper[2]. I have chosen the '4h' time frequency to be able to work with more granular data than daily prices and test how much an higher-frequency sampling of the data affects the performance of the fitted models when making out-of-sample predictions compared with the frequently used lower frequency sampling, such as daily observations, in the literature.

## 2.1 Theoretical Framework

The ARIMA(p,d,q), Autoregressive Integrated Moving Average, process first introduced by Box and Jenkins [13] is one of the most popular in the econometric literature and has been widely used for time series analysis and forecasting in discrete time. It is composed by two parts, the autoregressive one which is represented by the AR(p) process, and the moving average part, represented by the MA(q) process.

### 2.1.1 Stationarity

Both AR(p) and MA(q) are based on the assumption of stationarity, namely that both processes have a time-invariant distribution characterized by finite moments. A stochastic process, namely a ordered sequence of continuous random variables $Y_1, Y_2, \dots, Y_N$, $N \in T$ the set of times, is said to be strictly stationary if the moments of its distribution are constant for every point in time, that is, given two subsets of realizations of the stochastic process, $\{Y_t\}_{t=1}^{n}$ and $\{Y_t\}_{t=1+m}^{n+m}$, $\forall t \in N$, both sequences have the same probability distribution, thus the same moments, independently from time shifts 'm'. The assumptions of strict stationarity are too rigid for most applications because

[1]Application Programming Interface.

[2]https://github.com/sammchardy/python-binance

of the difficulty to verify their validity and often do not hold for financial time series. Instead of imposing that every moment is time invariant, weak stationarity assumes that only the first two moments of the distribution of a process are finite and time-independent, thus increasing the feasibility of verifying that the assumption holds in practice. For a weakly stationary process:

$$\mathrm{E}[Y_t] = \int_{-\infty}^{\infty} y\, f_t(y)\, dy = \mu < \infty\,, \qquad \forall t \tag{2.1}$$

$$Var[Y_t] = \mathrm{E}[Y - \mathrm{E}[Y]]^2 = \sigma^2 < \infty\,, \qquad \forall t \tag{2.2}$$

$$Cov[Y_t, Y_s] = \mathrm{E}[(Y_t - \mathrm{E}[Y_t])\,(Y_s - \mathrm{E}[Y_s])] = \gamma(|t-s|) < \infty\,, \qquad \forall t, s \tag{2.3}$$

Hence, the *autocovariance* function $\gamma(h)$ does not depend on time indexes themsleves, but only on time lags, namely the difference between two point $|t-s|$ in time where the stochastic process is being observed, this is why a weakly stationary process is often called covariance stationary. The autocovariance function can also be written as $\gamma(h) = Cov[Y_{t+h}, Y_t] = \mathrm{E}[(Y_{t+h} - \mu)\,(Y_t - \mu)]$ where $\gamma(h)$ depends only on the lag $h = |t+h-t|$ and the autocovariance function is symmetric, that is $\gamma(h) = \gamma(-h)$. The *autocorrelation* function is than derived as:

$$\rho(h) = \frac{Cov(Y_{t+h}, Y_t)}{\sqrt{Var(Y_{t+h})\,Var(Y_t)}} = \frac{\gamma(h)}{\gamma(0)} \tag{2.4}$$

By definition the autocorrelation function is a standardized measure of the linear relationship between sequences of variables drawn form the same stochastic process and shifted in time, such that $-1 \leq \rho(h) \leq 1$.

In equation (2.1), $f_t(y)$ is the marginal probability density function of the random variable $Y_t$ defined as $f_t(y) = \frac{\delta F_t(y)}{\delta y}$ where $F_t(y) = \Pr(Y_t \leq y)$ is the marginal distribution function. The benefit of stationarity assumptions is that there are less parameters to calculate in order to describe the distribution of a stochastic process, and given that all random variables have the same expected value $\mu$, the mean of the process can be accurately calculated by averaging trough the r.v.'s, $\overline{Y}$.

Hence, a stationary time series, which is a collection of equally spaced realizations of a stochastic process, $\{x_t\}_{t=0}^{n}$, chronologically ordered and with time index taking integer values, shows a well-defined behaviour when observing its evolution through time, that is, it is mean-reverting and randomly varying around a fixed level $\mu$. When the time series shows an evident time trend, departing form the initial mean level, the series is not stationary but behaving more like a random walk.

Even if the price of financial assets is best described as a continuous stochastic process, due to the method of sampling, that is, observing the price in equally spaced and discrete time points, often a time series is considered a discrete-time process and an approximation of a continuous stochastic one. In this context, the models employed in next sections assume that the values of the time series are observed in discrete time, reducing the number of parameters to estimate when fitting the distribution of the series, thus making calculations more feasible. Having said that, in order to take advantage of the statistical properties of continuous random variables, one can chose a specific method of transforming the series, for example continuous functions like the natural logarithm or the square root, or calculating the log returns of the corresponding prices, thus obtaining a series that can be modeled as a discrete-time continuous process [14].

The most basic stationary process is the white noise process defined as $\epsilon \sim WN(\mu, \sigma^2)$ with moments:

$$\mathrm{E}[\epsilon_t] = \mu\,, \qquad \forall t \tag{2.5}$$

$$Var[\epsilon_t] = \sigma_\epsilon^2\,, \qquad \forall t \tag{2.6}$$

$$Cov[\epsilon_t, \epsilon_s] = \begin{cases} \sigma_\epsilon^2\,, & \text{if } t = s \\ 0\,, & \text{if } t \neq s \end{cases} \tag{2.7}$$

Hence, a white noise process is a sequence of uncorrelated random variables and weakly stationary with autocovariance function $\gamma(h) = 0\ \forall h \neq 0$ and consequently $\rho(h) = 0\ \forall h \neq 0$ autocorrelation. When the expected value of the process is equal to zero, it is said to be distributed with mean zero and a finite variance with notation $\epsilon \sim WN(0, \sigma_\epsilon^2)$, additionally when $\epsilon_1, \epsilon_2, \ldots, \epsilon_t$ are i.i.d., independent and identically distributed random variables, the notation used is $\epsilon \sim i.i.d.\ WN(0, \sigma_\epsilon^2)$, and if $\epsilon_1, \epsilon_2, \ldots, \epsilon_t$ follow a specific distribution, such as the normal distribution, the noise variables are i.i.d. normally distributed $\epsilon \sim i.i.d.\ N(0, \sigma_\epsilon^2)$. These properties imply that when forecasting, the best linear predictor of a an i.i.d. white noise process is its mean because all the observations are uncorrelated and past observations do not provide any information about the future values of the process, indeed:

$$\mathrm{E}[\epsilon_{t+h} \mid \epsilon_t, \ldots, \epsilon_1] = \mu\,, \qquad \forall h > 0 \tag{2.8}$$

### 2.1.2 Autoregressive Process

An autoregressive process is based on the assumption that given a time series $\{y_t\}_{t=-\infty}^{+\infty}$ of correlated variables, the current observation $y_t$ can be expressed as a linear function of past observations, treated as explanatory variables, plus an unobservable noise term which adds randomness to the process. Thus an AR(p), where ‘p’ is the order of the process, can be defined as a weighted average of past values:

$$(y_t - \mu) = \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + \cdots + \phi_p(y_{t-p} - \mu) + \epsilon_t \tag{2.9}$$

Where $\phi_1, \phi_2, \ldots, \phi_p$ are constant and $\epsilon \sim WN(0, \sigma_\epsilon^2)$ is the ‘shock’ produced by new information at time t, thus $\epsilon_t$ is an uncorrelated white noise process by definition as the effects of new information on the current value $y_t$ can not be predicted by past shocks if they are truly unexpected, thus they need to be independent through time. The process may also be rewritten as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{2.10}$$

Where $c = \mu(1 - \phi_1 - \phi_2 - \cdots - \phi_p)$, and $\mu$ is the long run unconditional mean of the process.

Considering, for simplicity, a mean zero AR(1) process written as $y_t = \phi y_{t-1} + \epsilon_t$ by recursive iteration:

$$\begin{aligned} y_t &= \phi y_{t-1} + \epsilon_t = \phi(\phi y_{t-2} + \epsilon_{t-1}) + \epsilon_t \qquad (2.11)\\ &= \phi^2 y_{t-2} + \phi\epsilon_{t-1} + \epsilon_t \\ &\vdots \\ &= \phi^k y_{t-k} + \sum_{j=0}^{k-1} \phi^j \epsilon_{t-j} \qquad (2.12)\\ &\vdots \\ &= \sum_{j=0}^{\infty} \phi^j \epsilon_{t-j} \qquad (2.13)\end{aligned}$$

Where in (2.12) the first term on the right hand side converge to zero as $k \to \infty$ if $|\phi| < 1$ meaning that the dependency between current value $y_t$ and past values of the series decays to zero as the time distance increases. Additionally, the term $\sum_{j=0}^{k-1} \phi^j \epsilon_{t-j}$ shows that when k increases the weights of the shocks decay geometrically, namely that the effect on $y_t$ of distant past shocks is zero or negligible. (2.13) is also called $MA(\infty)$ representation of an AR(1) process and the latter is stationary if the condition $|\phi| < 1$ holds. Indeed, calculating the moments of AR(1) using (2.13) we obtain:

$$\mathrm{E}[y_t] = \sum_{j=0}^{\infty} \phi^j \mathrm{E}[\epsilon_{t-j}] = 0\,, \qquad \forall t \qquad (2.14)$$

$$Var[y_t] = Var[\sum_{j=0}^{\infty} \phi^j \epsilon_{t-j}] = \sigma_\epsilon^2 \sum_{j=0}^{\infty} \phi^{2j} = \frac{\sigma_\epsilon^2}{1-\phi^2}\,, \qquad \forall t \qquad (2.15)$$

$$Cov[y_{t+h}, y_t] = \mathrm{E}[(\sum_{j=0}^{\infty} \phi^j \epsilon_{t+h-j})\,(\sum_{i=0}^{\infty} \phi^i \epsilon_{t-i})] = \phi^h \frac{\sigma_\epsilon^2}{1-\phi^2}\,, \qquad \forall t \textit{ and } h \geq 0 \qquad (2.16)$$

$$Corr[y_{t+h}, y_t] = \frac{Cov[y_{t+h}, y_t]}{\sqrt{Var[y_{t+h}]Var[y_t]}} = \phi^h\,, \qquad \forall t \textit{ and } h \geq 0 \qquad (2.17)$$

as $\epsilon \sim WN(0, \sigma_\epsilon^2)$ and (2.15) is obtained by using the property of a geometrical series $\sum_{j=0}^{\infty} \phi^j = \frac{1}{1-\phi}$ if $|\phi| < 1$. From (2.17), the autocorrelation function of a stationary AR(1) process $\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \phi^h$ deacays geometrically to zero and follows an autoregressive process as $\rho(h) = \phi\rho(h-1)$ for $h \geq 1$.

Given the lag operator $L$ such that $L^h y_t = y_{t-h}$, the AR(p) process may be written in terms of the noise $\epsilon_t$:

$$(1 - \phi_1 L - \phi_1 L^2 - \cdots - \phi_p L^p) y_t = \epsilon_t \qquad (2.18)$$

or

$$\phi(L) y_t = \epsilon_t \qquad (2.19)$$

$$y_t = \phi(L)^{-1} \epsilon_t = \psi(L) \epsilon_t \qquad (2.20)$$

Where $\phi(L)$ is the autoregressive operator. The necessary condition In order for an autoregressive process to be written in its MA($\infty$) form is that the autoregressive polynomial needs to be invertible, the roots of $\phi(L)$ must lie outside the unit circle such that, $\phi(L) = 0$ only if $|L| > 1$. For example, an AR(1) of the form $y_t = \phi y_{t-1} + \epsilon_t$ can be rewritten in terms of the lag operator L as $y_t = \phi L y_t + \epsilon_t$ and rearranged such that $(1 - \phi L)y_t = \epsilon_t$, when $|\phi| < 1$, by using the property of geometric sequences:

$$y_t = (1 - \phi L)^{-1}\epsilon_t \tag{2.21}$$

$$\frac{1}{1 - \phi L} = 1 + \phi L + \phi^2 L^2 + \cdots + \phi^j L^j + \cdots , \qquad \forall\, |L| \leq 1 \tag{2.22}$$

$$y_t = \sum_{j=0}^{\infty} \psi_j \epsilon_{t-j} \tag{2.23}$$

Where $\psi_j = \phi^j$ and $\psi_0 = 1$. Equation (2.23) is called the Wold representation [15] of an infinite order AR(1) and the stationarity condition is $\sum_{j=0}^{\infty} \psi^2 < \infty$, coefficients are square summable or, a more stronger condition is the absolute summability of the coefficients $\sum_{j=0}^{\infty} |\psi| < \infty$ ensuring that $|\phi| < 1$ holds. The invertibility condition is fundamental for an autoregressive process to be causal, namely that the current value of the time series $y_t$ does not depend on future values $y_{t+1}, y_{t+2}, \ldots$, thus avoiding the look-ahead bias when fitting the model and building forecasts.

### 2.1.3 Random Walk

A basic example of a non-stationary autoregressive process is the random walk of the form $y_t = y_{t-1} + \epsilon_t$ where the autoregressive coefficient $\phi = 1$ in this case. A random walk without drift can be obtained by iterating backward t times as in (2.11), the process can be written as:

$$y_t = y_0 + \sum_{j=1}^{t} \epsilon_j \tag{2.24}$$

where $y_0$ is an arbitrary starting point and $\epsilon \sim WN(0, \sigma_\epsilon^2)$. Taking expectations on both sides to compute the moments of the process yields:

$$\mathrm{E}[y_t] = y_0 \tag{2.25}$$

$$Var[y_t] = \gamma_t(0) = 0 + \sum_{j=1}^{t} Var[\epsilon_j] = t\sigma^2 \tag{2.26}$$

$$Cov[y_t, y_{t-h}] = \gamma_t(h) = \mathrm{E}[(\sum_{j=1}^{t} \epsilon_j)\ (\sum_{i=1}^{t-h} \epsilon_i)] = min(t, t-h)\sigma^2 \,, \qquad \forall\, h > 0 \tag{2.27}$$

$$Corr[y_t, y_{t-h}]^2 = \rho_t^2(h) = \frac{\gamma_t^2(h)}{\gamma_t(0)\gamma_{t-h}(0)} = \frac{[(t-h)\sigma^2]^2}{[t\sigma^2][(t-h)\sigma^2]} = 1 - \frac{h}{t} \,, \qquad \forall\, h > 0 \tag{2.28}$$

The expected value of the process depends completely on the initial value $y_0$ and from (2.26) the variance of a non-stationary AR(1) increases linearly with time, therefore the value $y_t$ walks away form its mean, confirming that it is not mean-reverting. From (2.28), the autocorrelation function of a random walk process depends on time t, differently from a stationary AR, and one can notice that for t$\rightarrow \infty$ the process is perfectly linearly correlated and $\rho(h)$ does not fade away with time [16]. Moreover, for $|\phi| > 1$ the process has an explosive variance, the value $y_t$ detaches form its mean level exponentially faster thus making the calculation of the moments more difficult than the case where the stationary assumption holds.

### 2.1.4 Moving Average Process

Often, modeling a time series as an autoregressive process requires the estimation of too much parameters in order to obtain an adequate fit, with the risk of incurring in overfitting an AR model, that is why, to achieve a more parsimonious model, one has to fit an additional model that accounts for the noise terms and may be beneficial for the total computational complexity, reducing the number of parameters to be estimated. According to a moving average process MA(q), $y_t$ is expressed as a linear combination of past values of the noise term and for $\epsilon \sim WN(0, \sigma_\epsilon^2)$ takes the form:

$$y_t - \mu = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{2.29}$$

Where $\theta_1, \theta_2, \ldots, \theta_q$ are parameters. As for the autoregressive process, an MA(q) can be expressed in term of the moving average operator, such as:

$$\theta(L) = 1 + \theta_1 L + \theta_1 L^2 + \cdots + \theta_q L^q \tag{2.30}$$

$$y_t = \theta(L)\epsilon_t \tag{2.31}$$

Where $\mu$ has been set to zero. The process can also be expressed in its infinite form seen in (2.23) with $\psi_j = \theta_j$ for $j = 1, \ldots, q$, $\psi_0 = 1$ and $\psi_j = 0 \ \forall \ j > q$. A moving average process is always stationary, regardless of the values of the parameters theta, this can be shown by considering, for example, the moments of an MA(1) in the form $y_t = \epsilon_t + \theta\epsilon_{t-1}$, with $\mu = 0$ and assuming $\epsilon \sim WN(0, \sigma_\epsilon^2)$, the moments are:

$$\mathrm{E}[y_t] = 0 \tag{2.32}$$

$$\gamma(0) = \sigma_\epsilon^2(1 + \theta^2) \tag{2.33}$$

$$\gamma(1) = \mathrm{E}[(\epsilon_t + \theta\epsilon_{t-1})\ (\epsilon_{t+1} + \theta\epsilon_t)] = \theta\sigma_\epsilon^2 \tag{2.34}$$

$$\gamma(h) = 0\,, \qquad if\ h > 1 \tag{2.35}$$

$$\rho(1) = \frac{\theta}{1 + \theta^2} \tag{2.36}$$

$$\rho(h) = 0\,, \qquad if\ h > 1 \tag{2.37}$$

The autocovariance of MA(1) drops to zero when the lag between observations $y_t$ and $y_{t+h}$ is greater than 1, that is the order of the process, because when taking the expectation while computing an higher order autocovariance, for example $\gamma(2)$, there are no noise terms with a common time index. Consequently, the autocorrelation (2.36) summarize the behaviour of a moving average process of order 1 and will be useful for identifying which model and which order best fits the Bitcoin time series. Moreover, this behaviour is distinctive of all MA(q) processes and the autocorrelation cuts off $\forall |h| > q$.

Moreover, from equation (2.36) the same value of $\rho$ can be obtained with two different values of $\theta$, for example $\theta = 2$ and its reciprocal $\frac{1}{2}$ yield the same results and there is a problem of uniqueness and identification. To avoid this, in order to ensure uniqueness, the moving average process needs to be invertible, that is, $|\theta| < 1$ such that it admits and AR($\infty$) representation. Rewriting a MA(1) process in order to isolate the noise term at time t, $\epsilon_t = y_t - \theta\epsilon_{t-1}$, using the backward recursion as in (2.13) and substituting $\epsilon_{t-1} = y_{t-1} - \theta\epsilon_{t-2}$ iteratively, one can obtain the infinite autoregressive representation $\epsilon_t = \sum_{j=0}^{\infty}(-\theta)^j y_{t-j}$, where the term $(-\theta)^j \epsilon_{t-j}$ decays to zero $\forall$ $|\theta| < 1$ for $j \to \infty$ and is omitted. Considering the moving average polynomial representation of an MA(1) process $y_t = \theta(L)\epsilon_t$, if the invertibility condition is respected and one can write:

$$\theta(L) = 1 + \theta L$$

$$\pi(L) = \theta^{-1}(L) = \frac{1}{1+\theta L} = \sum_{j=0}^{\infty}(-\theta)^j L^j \,, \qquad if\ |\theta| < 1 \tag{2.38}$$

$$\pi(L) y_t = \epsilon_t \tag{2.39}$$

where (2.39) is the AR($\infty$) MA(1) representation.

### 2.1.5 ARMA process

An autoregressive moving average process, ARMA(p,q), is obtained by combining both AR(p) and MA(q) models and can be written as:

$$(y_t - \mu) = \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + \cdots + \phi_p(y_{t-p} - \mu) + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \tag{2.40}$$

or, in terms of polynomial operators:

$$(1 - \phi_1 L - \phi_1 L^2 - \cdots - \phi_p L^p)(y_t - \mu) = (1 + \theta_1 L + \theta_1 L^2 + \cdots + \theta_q L^q)\epsilon_t \tag{2.41}$$

$$\phi(L)(y_t - \mu) = \theta(L)\epsilon_t \tag{2.42}$$

with $\epsilon \sim WN(0, \sigma^2_\epsilon)$. From equation (2.41), when both the autoregressive and the moving average process share a common parameter, that is, when in both sides of the equation an autoregressive parameter $\phi$ is equal or extremely similar to a moving average paramenter $\theta$ it is a signal of over parametrization of the fitted model. In this case, paramenter redundancy can lead to a misinterpretation of the behaviour of the time series which observations are assumed to be correlated and fitted with an ARMA(p,q) model, when in reality if one cancels out the common terms from both sides of the equation, the underlying process behaves more like a random walk and additional explanatory variables coefficients would not be statistically significant. To avoid this, in order to have a well-defined ARMA process, there are two conditions, one for the AR part and one for the MA part, that must hold for the ARMA(p,q) model to be causal and invertible.

First, the causality condition depends on the AR part and, given an ARMA(p,q) process, the latter is said to be causal if it admits an infinite moving average representation:

$$y_t = \sum_{j=0}^{\infty} \Psi_j \epsilon_{t-j} = \Psi(L)\epsilon_t \tag{2.43}$$

Where $\Psi_0 = 1$, $\sum_{j=0}^{\infty} |\Psi_j| < \infty$ and the autoregressive part respects the condition for AR(p) process to be causal and the associated autoregressive polynomial to be invertible as defined earlier. Equation (2.43) follow the approach proposed by Box and Jenkins [13] for time series modeling and $\Psi(L) = \sum_{j=0}^{\infty} \Psi_j L^j = \frac{\theta(L)}{\phi(L)}$ for $|L| \leq 1$.

The invertibility condition depends on the moving average part of the process, that is:

$$\Pi(L)y_t = \sum_{j=0}^{\infty} \Pi_j y_{t-j} = \epsilon_t \tag{2.44}$$

Where $\Pi_0 = 1$, $\sum_{j=0}^{\infty} |\Pi_j| < \infty$ and $\Pi(L) = \sum_{j=0}^{\infty} \Pi_j L^j = \frac{\phi(L)}{\theta(L)}$ for $|L| \leq 1$. As before, an ARMA(p,q) is invertible if and only if the roots of the MA(q) polynomial $\theta(L)$ all lie outside the unit circle, $|L| > 1$.

As regards the moments of an ARMA(p,q) process, studying a simpler ARMA(1,1) of the form $y_t = \phi y_{t-1} + \epsilon_t + \theta\epsilon_{t-1}$, assuming $\mu = 0$ and $\epsilon \sim WN(0, \sigma_\epsilon^2)$, one can observe its behaviour:

$$\mathrm{E}[y_t] = \sum_{j=0}^{\infty} \Psi_j \mathrm{E}[\epsilon_{t-j}] = 0 \tag{2.45}$$

by multiplying both sides of $y_t = \phi y_{t-1} + \epsilon_t + \theta\epsilon_{t-1}$ by $\epsilon_t$ and taking expectations [17] :

$$Cov(y_t, \epsilon_t) = \mathrm{E}[y_t\epsilon_t] - \mathrm{E}[y_t]\,\mathrm{E}[\epsilon_t] = \mathrm{E}[\phi y_{t-1}\epsilon_t + \epsilon_t^2 + \theta\epsilon_{t-1}\epsilon_t] = \sigma_\epsilon^2 \tag{2.46}$$

given (2.45) and $\mathrm{E}[\epsilon_t] = 0$. The autocovariance function is defined as:

$$\gamma(0) = \mathrm{E}[y_t y_t] = \frac{(1 + \theta^2 + 2\phi\theta)\sigma_\epsilon^2}{1 - \phi^2} \tag{2.47}$$

and the autocovariance for lag $|h| = 1$ is:

$$\begin{aligned}
\gamma(1) &= \mathrm{E}[y_t y_{t-1}] && (2.48)\\
&= \mathrm{E}[(\phi y_{t-1} + \epsilon_t + \theta\epsilon_{t-1})\ (\phi y_{t-2} + \epsilon_{t-1} + \theta\epsilon_{t-2})] && (2.49)\\
&= \frac{(1 + \theta\phi)(\theta + \phi)\sigma_\epsilon^2}{1 - \phi^2} && (2.50)
\end{aligned}$$

hence, the autocorralation function is given by:

$$\rho(1) = \frac{(1 + \theta\phi)(\theta + \phi)}{(1 + \theta^2 + 2\phi\theta)} \tag{2.51}$$

which for $|h| \geq 2$ follows an autoregressive process and behaves in the same way as in AR(1) as the effect of the MA(1) part on the correlation of the process cuts off after a time lag greater than 1:

$$\rho(h) = \phi\rho(h-1)\,, \qquad for\ |h| \geq 2 \tag{2.52}$$

As regards the autocovariance and autocorrelation functions of the general ARMA(p,q) process, they reflects the behaviour of both $\gamma(h)$ and $\rho(h)$ of AR(p) and MA(q) processes combined together. Considering the general MA(q) first, in its compact form, assuming $\theta_0 = 1$ and $\epsilon \sim WN(0, \sigma_\epsilon^2)$, taking expectations on both sides [18] :

$$\mathrm{E}[y_t] = \mathrm{E}[\theta(L)\epsilon_t] = \sum_{j=0}^{q} \theta_j \mathrm{E}[\epsilon_{t-j}] = 0 \tag{2.53}$$

then, with the same considerations made in section 2.1.4 for the calculation of the autocovaricance function of an MA(1), the autocovariance of the general MA(q) is computed as:

$$\gamma(h) = \mathrm{E}[(\sum_{j=0}^{q} \theta_j \epsilon_{t+h-j})\,(\sum_{i=0}^{q} \theta_i \epsilon_{t-i})] \tag{2.54}$$

$$= \begin{cases} \sigma_\epsilon^2 \sum_{j=0}^{q-h} \theta_j \theta_{j+h}\,, & \text{if } 0 \leq h \leq q \\ 0\,, & \text{if } h > q \end{cases} \tag{2.55}$$

dividing by $\gamma(0)$, the autocorrelation function is:

$$\rho(h) = \begin{cases} \dfrac{\sum_{j=0}^{q-h} \theta_j \theta_{j+h}}{1 + \theta_1^2 + \cdots + \theta_q^2}\,, & \text{if } 1 \leq h \leq q \\ 0\,, & \text{if } h > q \end{cases} \tag{2.56}$$

The autocovariance and the ACF of an AR(p) are easier to obtain because they follow an autoregressive process of order p:

$$\gamma(h) = \phi_1 \gamma(h-1) + \cdots + \phi_p \gamma(h-p)\,, \qquad h \geq p \tag{2.57}$$

and dividing both sides by $\gamma(0)$:

$$\rho(h) = \phi_1 \rho(h-1) + \cdots + \phi_p \rho(h-p)\,, \qquad h \geq p \tag{2.58}$$

Finally, considering both the compact form of an ARMA(p,q) in terms of the infinite polynomial $\Psi(L)$, $y_t = \sum_{j=0}^{\infty} \Psi_j \epsilon_{t-j}$ and the ordinary form, where $\mu$ is assumed to be zero and the errors are white noise as usual, $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$, the autocovariance and ACF of an ARMA(p,q) process are obtained from:

$$\gamma(h) = \mathrm{E}[(\sum_{j=1}^{p} \phi_j y_{t+h-j} + \sum_{j=0}^{q} \theta_j \epsilon_{t+h-j})\; y_t] \tag{2.59}$$

$$= \sum_{j=1}^{p} \phi_j \gamma(h-j) + \sigma_\epsilon^2 \sum_{j=h}^{q} \theta_j \Psi_{j-h}\,, \qquad for\ 0 \leq h < max(p, q+1) \tag{2.60}$$

$$= \phi_1 \gamma(h-1) + \cdots + \phi_p \gamma(h-p)\,, \qquad for\ h \geq max(p, q+1) \tag{2.61}$$

where in (2.60) the covariance between $\epsilon_{t+h-j}$ and $\sum_{k=0}^{\infty} \Psi_k \epsilon_{t-k}$ starts taking values when $k = j-h$ as the first covariate can be expressed equivalently as $\epsilon_{t-(j-h)}$. The ACF can be obtained by dividing both sides of (2.60) and (2.61) with $\rho(h)$. From the equations above, the autocovariance, and consequently the ACF, of an ARMA(p,q) depends on both AR(p) and MA(q) terms if the lag $h$ is less than the order of the moving average process, when $h$ becomes greater than $q$, $\gamma(h)$ and $\rho(h)$ are completely dominated by the autoregressive part of the model and behave like a AR(p) only process, as expected.

The partial autocorrelation function, abbreviated PACF, is useful for estimating the order of dependence for a stochastic process, that is, estimating the correlation between two observations spaced by a certain time lag $h$, partialling out the linear effect of the values observed between them from the estimate. As seen before, in the case of the MA(q) model, the fact that its ACF cuts off after the lag $h \geq q$, and is defined when $h = q$ as $\theta_0 \neq 0$, the order of dependence is readily obtained, but for an AR(p) process the autocovariance and ACF do not cut off after $h > p$ but decays as $h$ grows in magnitude. Therefore, the idea is to regress both observations against the intermediate variables between them that affect their linear correlation, and removing these effects when computing the correlation. Hence, the PACF of a stationary process $y_t$ is defined as:

$$\phi_{hh} = Corr(y_{t+h} - \hat{y}_{t+h},\, y_t - \hat{y}_t)\,, \qquad h > 1 \tag{2.62}$$

where $\phi_{11} = \rho(1)$, $\hat{y}_{t+h}$ and $\hat{y}_t$ are respectively the regression of $y_{t+h}$ on $y_{t+h-1}, y_{t+h-2}, \ldots, y_{t+1}$ and $y_t$ on $y_{t+1}, y_{t+2}, \ldots, y_{t+h-1}$, such that the effect of the terms $\{y_{t+1}, \ldots, y_{t+h-1}\}$ are removed when computing the PACF. For a stationary AR(p) process, $y_{t+h} = \sum_{j=1}^{p} \phi_j y_{t+h-j} + \epsilon_{t+h}$ and $\hat{y}_{t+h} = \sum_{j=1}^{p} \phi_j y_{t+h-j}$, such that when $h > p$ the PACF is, $Corr(y_{t+h} - \hat{y}_{t+h},\, y_t - \hat{y}_t) = Corr(\epsilon_{t+h}, y_t - \hat{y}_t) = 0$. Thus, the PACF of an autoregressive process behaves in the opposite way of its ACF, that is, it cuts off at a certain lag, depending on the order of the process, and can be used for estimating the order of dependence between the values of the time series. On the other hand, an MA(q) process does not admit a finite AR representation but an infinite one when it is invertible, this makes impossible to remove the intermediary values dependent on $y_t$ and $y_{t+h}$ because they are infinitely many, hence the PACF of a moving average process never cuts off after a lag equal to $q$ but decays in the same way the ACF of an AR(p) does, thus tailing off in contrary to the MA(q) ACF. As regards a causal and invertible ARMA(p,q), it always admits an infinite AR representation such that its PACF always tails off like in the moving average case as discussed above.

### 2.1.6 ARIMA model

Financial time series are notoriously non-stationary, that is, they rarely show a mean-reverting behaviour around the same fixed mean level, sometimes their evolution show momentum in some periods, or instead they follow a non-stationary trend that has a time-varying mean and depart, linearly or non-linearly, from the initial mean level. In these cases, the ARIMA, autoregressive integrated moving average, are employed to model non-stationary time series by applying the so called differencing operator '$\Delta$' to the whole series, that is taking the difference of the observations sequentially, as many time as it is required to transform the series in a stationary one. For example, if the time series is composed by log prices, fitting an ARIMA(p,1,q) will results in fitting and ARMA(p,q) on the log returns of the series. In order to show how much differencing a process can transform it, take as an example the simple random walk process with zero mean and stationary noise term defined as $y_t = y_{t-1} + \epsilon_t$, the first order difference transforms the random walk in a stationary process, that is, $y_t - y_{t-1} = \Delta y_t = \epsilon_t$. A process that presents a non-stationary trend component, for example $y_t = \mu_t + \epsilon_t$, where $\epsilon_t$ is a zero mean stationary process and $\mu_t = \beta_0 + \beta_1 t$ is a deterministic linear trend such as $y_t$ is non-stationary because the mean of the process varies with time, $\mathrm{E}[y_t] = \beta_0 + \beta_1 t$, applying a first order difference leads to $\Delta y_t = y_t - y_{t-1} = \beta_0 + \beta_1 t - \beta_0 - \beta_1(t-1) + \epsilon_t - \epsilon_{t-1} = \beta_1 + \Delta\epsilon_t$ which is a stationary process. Differencing can be applied repetitively, for example a second order difference of the process above is defined as $\Delta^2 = \Delta(\Delta y_t)$. The inverse process of differencing is integrating and a stochastic process *I(d)* is called integrated of order *d*, therefore it needs to be differenced *d* times in order to be stationary and a stationary process, for example a white noise, is defined as *I(0)*.

An ARIMA(p,d,q) is defined as:

$$\phi(L)\Delta^d y_t = \phi(L)(1-L)^d y_t = \theta(L)\epsilon_t \tag{2.63}$$

the intercept is missing because often differencing a series has a de-meaning effect, still, if $\mathrm{E}[\Delta^d y_t] = \mu \neq 0$ the process becomes:

$$\phi(L)\Delta^d y_t = \phi(L)(1-L)^d y_t = c + \theta(L)\epsilon_t \tag{2.64}$$

where $c = \mu(1 - \phi_1 - \phi_2 - \cdots - \phi_p)$.

## 2.2 Conditional Heteroscedasticity Modeling

The models considered so far are all built on the assumption of constant conditional variance, indeed the innovation term has always been assumed to follow a white noise process with mean zero and fixed volatility $\sigma^2$. The assumption of homoscedasticity, that is, the conditional variance of a stochastic process depends on the noise term which follows a constant variance distribution, falls short when modeling financial time series that often are characterized by periods of consistently increasing, or decreasing volatility, showing clear signs of time-varying volatility. Indeed, when fitting an ARMA(p,q) model to a time series, the residuals of the model can exhibit an heteroscedastic behaviour making necessary to fit a model that can explain the effect of a time varying volatility on the conditional mean of the underlying process.

Additionally, in financial markets, there are periods of very high volatility of asset returns, for example due to a new piece of information dissipating through participants, that are usually followed by other periods of high volatility before the new information is incorporated in the price of the asset or when the temporary inefficiency is arbitraged out. This phenomenon is called volatility clustering, spikes in volatility that are grouped together and exhibit temporal dependency between each other, and can be detected from the residuals of the fit of a constant conditional variance model.

### 2.2.1 ARCH Process

The first attempt to model a time-varying conditional variance was performed by Engle [19] who presented the Autoregressive Conditional Heteroscedasticity, ARCH model. The general form of an ARCH(p), where $p$ indicates the order of the process, is:

$$a_t = \epsilon_t \sigma_t \tag{2.65}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i a_{t-i}^2 \tag{2.66}$$

Consider the simple ARCH(1), defined as:

$$a_t = \epsilon_t \sigma_t \tag{2.67}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 \tag{2.68}$$

with the constraints $\alpha_0 > 0$ and $\alpha_1 \geq 0$ because the variance cannot be negative, the error term $\epsilon_t$ is assumed to be i.i.d. Gaussian white noise with mean zero and unit variance:

$$\mathrm{E}[\epsilon_t \mid F_{t-1}] = 0 \tag{2.69}$$

$$Var[\epsilon_t \mid F_{t-1}] = 1 \tag{2.70}$$

where $F_{t-1}$ is the information set available at time $t-1$, that is $F_{t-1} = \{\epsilon_{t-1}, \dots\}$. From (2.67), the ARCH(1) model is based on the assumption that, if the returns of an asset can be modeled as $r_t = \mu_t + a_t$ with the conditional mean component $\mu_t$ that for example follows an ARMA(p,q) process, the shock term $a_t = r_t - \mu_t$ is modeled as an ARCH(1) with conditional mean equal to zero as $\mathrm{E}[a_t \mid F_{t-1}] = 0$ so that $a_t$ is uncorrelated with $r_t$ but has non-constant conditional variance (2.68) that is dependent on past values of the shocks. This characteristic of the ARCH(1) makes the latter possible to be used for modeling a time-varying variance and account for volatility clustering when modeling $r_t$. Moreover, the main difference between an ARCH(1) process and an AR(1) can be noticed by taking the square on both sides of (2.65), obtaining $a_t^2 = \epsilon_t(\alpha_0 + \alpha_1 a_{t-1}^2)$, $a_t^2$ follows a similar evolution to an AR(1) but now the shocks terms are assumed to be multiplicative and not additive.

By the law of iterated expectations, taking the expectation of the conditional mean of an ARCH(1) process, the unconditional mean of $a_t$ is still zero as $\mathrm{E}[a_t] = \mathrm{E}[\mathrm{E}(a_t \mid F_{t-1})] = \mathrm{E}[\mathrm{E}(\epsilon_t \sigma_t)] = 0$ because the noise term is independent of $a_t$. Recalling that $\mathrm{E}[\epsilon_t^2] = 1$, the conditional variance of $a_t$ can be obtained as:

$$\begin{aligned} Var[a_t \mid F_{t-1}] &= \mathrm{E}[a_t^2 \mid F_{t-1}] && (2.71)\\ &= \mathrm{E}[\epsilon_t^2(\alpha_0 + \alpha_1 a_{t-1}^2) \mid F_{t-1}] && (2.72)\\ &= [\alpha_0 + \alpha_1 a_{t-1}^2 \,\mathrm{E}(\epsilon_t^2 \mid F_{t-1})] && (2.73)\\ &= \alpha_0 + \alpha_1 a_{t-1}^2 && (2.74)\end{aligned}$$

Again, by the law of iterated expectations, the unconditional variance of an ARCH(1) process can be computed by taking the expectation of (2.74):

$$\begin{aligned} \gamma_a(0) = \mathrm{E}[\mathrm{E}(a_t^2 \mid F_{t-1})] &= \mathrm{E}[\alpha_0 + \alpha_1 a_{t-1}^2] && (2.75)\\ &= [\alpha_0 + \alpha_1 \,\mathrm{E}(a_{t-1}^2)] && (2.76)\\ &= \frac{\alpha_0}{(1-\alpha_1)} && (2.77)\end{aligned}$$

The last result shows that the necessary condition for an ARCH(1) process to have a finite unconditional variance is $0 \leq \alpha_1 < 1$ , such that the process is stationary and $\mathrm{E}(a_{t-1}^2) = Var[a_{t-1}] = Var[a_t] = \gamma_a(0)$ so that one can rearrange the terms in (2.76) to get (2.77). Given that the conditional mean of $a_t$ is zero, the best expectation of $a_t$ given its past its zero, the correlation for any lag greater than zero will be zero, so $\rho_a(h) = 0$ and an ARCH(1) process is the perfect example of an uncorrelated but dependent process and the dependency stems from its conditional variance. Indeed, as can be seen from (2.74), the conditional variance is built in a way such that it functions as a proxy between present and past values of the shocks, a sudden increase, or decrease, of $a_{t-1}$ is propagated to $a_t$ through $\sigma_t^2$ such that it helps explaining the dependence of shocks in a volatility clustering environment and keeps reacting to the magnitude of past shocks as their values changes, even if, by definition, the value of the conditional variance converge to the unconditional variance in the long-run.

By computing the the fourth moment of an ARCH(1) process, one can derive the formula for the kurtosis of its distribution in terms of $\alpha_1$:

$$\kappa = \frac{\mathrm{E}[a_t^4]}{[Var(a_t)]^2} = 3\frac{1-\alpha_1^2}{1-3\alpha_1^2}$$

with the constraint that $0 \leq \alpha_1 < \frac{1}{3}$. From this results, it can be noticed that the kurtosis of the ARCH(1) distribution is higher than 3, which is value of the kurtosis of a Normal distribution, that is why the ARCH(1) model is useful for modeling series that exhibit fatter tails than the Normal and are more outliers-prone.

Finally, for a stationary ARCH(1), the ACF expressed in terms of $a_t^2$ behaves like the ACF of an AR(1) process, that is, $\rho_{a^2}(h) = \alpha^{|h|}$, $\forall\ h$ which decays geometrically as in the autoregressive case.

### 2.2.2 GARCH Process

The ARCH(p) model has some limitations, it treats positive and negative shocks as equally impacting the conditional volatility as the value of past shock in the conditional volatility formula is squared, when in reality, for financial time series, usually the returns of an asset react differently from bad news than from positive news and often there is a clear distinction of the magnitude of fluctuations caused by different factors. Additionally, the constraints imposed for the parameters in order for the process to have a finite fourth moment make calculations cumbersome when an higher order ARCH model is chosen, making difficult to account for returns' fat tails. Due to the way it is designed, ARCH will tend to overpredict the volatility when an higher than average and isolated value of $a_t^2$ occurs because it is directly fed into $\sigma_t^2$ and the value of the volatility will quickly stack up and converge to the long-run level rather quickly as well.

To overcome some of these shortcomings, bollerslev [20] presented the Generalized Autoregressive Conditional Heteroscedasticity model, abbreviated GARCH with the main aim to make the conditional volatility more persistent and reactive with respect to sudden shocks. The general GARCH(p,q) form is:

$$a_t = \epsilon_t \sigma_t \tag{2.78}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i a_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{2.79}$$

where $\epsilon \sim i.i.d.\ N(0,1)$, $\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_j \geq 0$ and in order for the process to have a finite unconditional variance $\sum_{i=0}^{max(p,q)}(\alpha_i + \beta_i) < 1$ under the condition that the alpha terms are zero when $i$ is greater then the order of the ARCH part of the model, that is $p$, and beta terms are zero when $i > q$. The stationarity condition of a GARCH(p,q) process looks very similar to the one related to an ARMA(p,q) process, indeed it is possible to obtain an ARMA representation of GARCH by defining $\eta_t = a_t^2 - \sigma_t^2$ to be an uncorrelated white noise process with mean zero such that it is uncorrelated with $a_t$ but it is not i.i.d. in order to ensure conditional heteroscedasticity for the $a_t$ process. Then, by using this relation, the ARMA representation of a GARCH(p,q) model is given by:

$$a_t^2 = \alpha_0 + \sum_{i=1}^{max(p,q)} (\alpha_i + \beta_i) a_{t-i}^2 + \eta_t - \sum_{j=1}^{q} \beta_j \eta_{t-j} \tag{2.80}$$

and the unconditional mean of an ARMA process is defined as:

$$\mathrm{E}[a_t^2] = \frac{\alpha_0}{1 - \sum_{i=1}^{max(p,q)}(\alpha_i + \beta_i)} \tag{2.81}$$

which is the unconditional variance of a GARCH(p,q) process and it is the long run level to which the conditional variance will converge to, provided that $\sum_{i=0}^{max(p,q)}(\alpha_i + \beta_i) < 1$.

From (2.79) it can be noticed that, differently from an ARCH only process, the lagged values of both the shocks and the conditional variance influence the present value of the conditional volatility such that a GARCH(p,q) model can better represent volatility clusters as the iterative effects of both $a_{t-1}^2$ and $\sigma_{t-1}^2$ on $\sigma_t^2$ make the effect of sudden shocks more persistent and the model reacts faster to volatility spikes. Finally, a GARCH(p,q) has a kurtosis greater than 3 as well, so it can be used to model a process with heavy tails.

## 2.3 Parameter Estimation & Forecasting

In order to fit the models to the Bitcoin time series and estimate the parameter of the distribution I have used the *rugarch* [3] package in R and amongst the available approaches to parameter estimation I have chosen the Maximum Likelihood method. The latter is based on the assumption that, given a sequence of i.i.d. random variables $y_1, \ldots, y_N$, they are drawn from a known parametric distribution that can be described by a vector of parameters $\boldsymbol{\omega}_0 \in \Omega$, and since that they are i.i.d., their joint probability density can be written as the product of the marginal densities $f(y_1, \ldots, y_N; \boldsymbol{\omega}_0) = \prod_{t=1}^{N} f(y_t; \boldsymbol{\omega}_0)$. In reality, the true values of the parameters are unknown and rewriting the joint density as $f(y_1, \ldots, y_N; \boldsymbol{\omega})$, the latter describe how likely the $y_N$ are realizations of the known probability distribution given a fixed set of parameters values $\boldsymbol{\omega}$. If one keeps the $y_N$ fixed and let the parameter values vary, the likelihood function describes the likelihood that the distribution of the given sequence of observations is actually described by the parameter vector $\boldsymbol{\omega}$ and the goal is to maximize this probability by finding a set of parameters that best approximate the population parameters $\boldsymbol{\omega}_0$. The likelihood is defined as:

$$L(\boldsymbol{\omega}; y_N) = \prod_{t=1}^{N} f(y_t; \boldsymbol{\omega}) \tag{2.82}$$

and the population parameters maximize (2.82) such that:

$$\boldsymbol{\omega}_0 = \underset{\boldsymbol{\omega}_0 \in \Omega}{argmax}\ \mathrm{E}[L(\boldsymbol{\omega}; y_N)] \tag{2.83}$$

Often, it is better to work with sums instead of products when maximizing a function so by taking the logarithm of (2.82) one obtains the log-likelihood function:

$$\ell(\boldsymbol{\omega}; y_N) = log[L(\boldsymbol{\omega}; y_N)] = \sum_{t=1}^{N} log\, f(y_t; \boldsymbol{\omega}) \tag{2.84}$$

The maximum likelihood estimates, abbreviated MLE, of the parameter vector are the values that maximize the sample average of the log-likelihood:

$$\hat{\boldsymbol{\omega}}_{MLE} = \underset{\boldsymbol{\omega} \in \Omega}{argmax}\ \mathrm{E}_N[\ell(\boldsymbol{\omega}; y_N)] \tag{2.85}$$

where $\mathrm{E}_N[\ell(\boldsymbol{\omega}; y_N)] = \frac{1}{N}\sum_{t=1}^{N} \ell(\boldsymbol{\omega}; y_t)$. When fitting an ARIMA with GARCH innovations to a time series, the assumption is that the the observations are correlated with their own lagged values and the model is fitted to try to explain this serial correlation. In this case, the variables $y_N$ are

[3] https://cran.r-project.org/web/packages/rugarch/rugarch.pdf

not i.i.d. anymore so their joint correlation cannot be written as the product of their marginal densities but it involves the product of conditional densities, indeed:

$$f(y_1, \dots, y_N) = f(y_N \mid y_{N-1}, \dots, y_1)\, f(y_{N-1}, \dots, y_1) \tag{2.86}$$

and by repeated substitutions, the joint density of dependent variables can be written as:

$$f(y_1, \dots, y_N) = f(y_1) \prod_{t=2}^{N} f(y_t \mid y_{t-1}, \dots, y_1) = f(y_1) \prod_{t=2}^{N} f(y_t \mid F_{t-1}) \tag{2.87}$$

Hence, in the case of dependent variables, in order to make calculations easier, usually the term $f(y_1)$ in (2.86) is dropped form the joint density and parameter estimation is be carried by maximizing the conditional likelihood, that is, conditioning on the first observations.

In the case of ARIMA(P,d,Q)-GARCH(p,q), where I have used uppercase letters for the orders of the ARIMA model to distinguish them from the GARCH ones, recall that:

$$\phi(L)\Delta^d(y_t - \mu) = \theta(L)a_t \tag{2.88}$$

$$a_t = \epsilon_t \sigma_t \tag{2.89}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i a_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{2.90}$$

where, $\epsilon \sim i.i.d.\,(0,1)$ and from (2.88) now the innovation terms are modeled as a GARCH process with a time-warying conditional variance $\sigma_t^2$. The first equation concerning the conditional mean of the process can also be written in the extended form as $\tilde{y}_t = \sum_{i=1}^{P} \phi_i \tilde{y}_{t-i} + a_t + \sum_{j=1}^{Q} \theta_j a_{t-j}$, where $\tilde{y}_t = \Delta^d(y_t - \mu)$, such that the distribution of the observations is $\tilde{y}_t \mid F_{t-1} \sim N(\sum_{i=1}^{P} \phi_i \tilde{y}_{t-i} + \sum_{j=1}^{Q} \theta_j a_{t-j},\, \sigma_t^2)$ because, conditional on the information set, only the expected value of $a_t$ is zero as at time $t-1$ the shocks are unobservable whereas the values of $a_{t-1}, \dots$ are in $F_{t-1}$. To ease of notation I will use $\mu_t$ when referring to the conditional mean of $y_t$, such that $y_t \mid F_{t-1} \sim N(\mu_t,\, \sigma_t^2)$ and the process has the equivalent form:

$$y_t = \mu_t + a_t \tag{2.91}$$

To estimate the parameter vector $\boldsymbol{\omega} = ((\mu, \phi_1, \dots, \phi_P, \theta_1, \dots, \theta_Q)', (\alpha_0, \alpha_1, \dots, \alpha_q, \beta_1, \dots, \beta_q)')'$ [4] the likelihood function of the data is needed and it will depend on the assumption of the distribution of $\epsilon_t$. In general, define $\epsilon_t = \frac{a_t}{\sigma_t}$ to be the standardized innovations of the process with a probability density function $f(\epsilon_t; \boldsymbol{\omega}, \boldsymbol{\eta})$, where $\boldsymbol{\eta}$ are additional parameters like the shape in case the innovations are assumed to follow a different distribution than the standard normal, since $\epsilon_t$ are unobservable, the conditional pdf of the data can be obtained by applying the rule of the change of variable, that is, when $f(\epsilon_t)$ is known [21] :

$$f(y_t; \boldsymbol{\omega}, \boldsymbol{\eta} \mid F_{t-1}) = f_\epsilon(h^{-1}(y_t, \boldsymbol{\omega}); \boldsymbol{\eta}) \left| \frac{\delta h^{-1}(y_t, \boldsymbol{\omega})}{\delta y_t} \right| \tag{2.92}$$

$$= f(\epsilon_t; \boldsymbol{\omega}, \boldsymbol{\eta}) \frac{1}{\sigma_t} \tag{2.93}$$

[4]where $\alpha_0 > 0$, $\alpha_1, \dots, \alpha_q \geq 0$, $\beta_1, \dots, \beta_q \geq 0$

where $y_t = h(\epsilon_t, \boldsymbol{\omega}) = \mu_t(\boldsymbol{\omega})+\sigma_t(\boldsymbol{\omega})$ and $\epsilon_t = h^{-1}(y_t, \boldsymbol{\omega}) = \frac{y_t-\mu_t(\boldsymbol{\omega})}{\sigma_t(\boldsymbol{\omega}}$. The conditional log-likelihood for $y_t$ is then obtained by taking the logs of (2.93), that is:

$$\ell_t(\boldsymbol{\omega}, \boldsymbol{\eta}; y_t \mid F_{t-1}) = log\, f(\epsilon_t; \boldsymbol{\omega}, \boldsymbol{\eta}) - \frac{1}{2} log\, \sigma_t^2(\boldsymbol{\omega}, \boldsymbol{\eta}) \tag{2.94}$$

When the standardized innovations are assumed to follow a normal distribution, $\epsilon_t \sim N(0, 1)$, their pdf is $f(\epsilon_t; \boldsymbol{\omega}) = \frac{1}{\sqrt{2\pi}} exp(-\frac{\epsilon_t^2}{2})$ and the conditional log-likelihood of the entire sample is:

$$\ell(\boldsymbol{\omega}; y_N \mid F_{N-1}) = -\frac{N}{2} log(2\pi) - \frac{1}{2}\sum_{t=1}^{N} log(\sigma_t^2(\boldsymbol{\omega})) - \frac{1}{2}\sum_{t=1}^{N} \frac{a_t^2(\boldsymbol{\omega})}{\sigma_t^2(\boldsymbol{\omega})} \tag{2.95}$$

Which is maximized through an iterative approach in order to find the MLE $\hat{\boldsymbol{\omega}}$. In order to start the recursion and initialize the conditional likelihood, the *rugarch* package sets the conditional least squares estimates as the initial values of $a_t^2$. Similarly, *rugarch* offers different methods to compute the initial values of the conditional variance $\sigma_t^2$ since the latter is unobservable at the start of the recursion, and following the indications of Fiorentini et al. [22], I have opted to set their initial values equal to the sample average of the squared residuals $a_s^2$ obtained as least squares estimates through a preliminary regression of $y_t$ on its lagged values as regressors. The vector of maximum likelihood estimates $\hat{\boldsymbol{\omega}}$ is then found with a numerical constrained optimization approach by solving the first order conditions for the score vector $S(\hat{\boldsymbol{\omega}}; y_N)$ containing the gradients of the conditional log-likelihood with respect to the parameters $\omega_0, \omega_1, \ldots$, namely, setting the partial derivatives of $\ell(\boldsymbol{\omega}, \boldsymbol{\eta}; y_N \mid F_{N-1})$ with respect to the parameters $\boldsymbol{\omega}$ equal to zero:

$$S(\hat{\boldsymbol{\omega}}; y_N) = \mathrm{E}_N \left[\frac{\delta\ell(\boldsymbol{\omega}; y_N \mid F_{N-1})}{\delta\boldsymbol{\omega}}\right]_{\boldsymbol{\omega}=\hat{\boldsymbol{\omega}}} = 0 \tag{2.96}$$

provided that the conditional log-likelihood is differentiable in $\boldsymbol{\omega}$, $\forall \boldsymbol{\omega} \in \Omega$. From probability theory, the MLE $\hat{\boldsymbol{\omega}}$ is an unbiased estimator of the population parameter vector $\boldsymbol{\omega}_0$ and it has been proven that when $N \to \infty$ by the central limit theorem and the law of large number, the maximum likelihood estimator is consistent and asymptotically normally distributed as $\sqrt{n}(\hat{\boldsymbol{\omega}} - \boldsymbol{\omega}_0) \overset{d}{\to} N(0, I(\boldsymbol{\omega}_0)^{-1})$, where $I(\boldsymbol{\omega}_0)^{-1}$ is the inverse of the Fisher information matrix computed as $I(\boldsymbol{\omega}_0) = Var[S(\boldsymbol{\omega}_0; y_N)] = -\mathrm{E}\left[\frac{\delta^2\ell(\boldsymbol{\omega}_0; y_N)}{\delta\boldsymbol{\omega}\, \delta\boldsymbol{\omega}'}\right]_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}$ which is the negative hessian matrix containing the second order partial derivatives of the log-likelihood with respect to the population parameters. Since the latter is unobservable in practice as $\boldsymbol{\omega}_0$ is unknown, for a sample large enough, $\hat{\boldsymbol{\omega}}$ is an efficient estimator of the real parameters and by the LLN the observed version of the inverse information matrix relative to the sample conditional log-likelihood and computed with respect to the MLEs converges asymptotically to $I(\boldsymbol{\omega}_0)^{-1}$ which is called the Cramer-Rao [23] [24] lower bound for the asymptotic variance-covariance matrix of the maximum likelihood estimator. The observed information matrix is essential to evaluate the accuracy of the maximum likelihood estimators and is used to construct the confidence intervals for the estimated parameters using as standard errors the inverse of the diagonal terms of $I(\hat{\boldsymbol{\omega}})$.

Usually, the assumption that the standardized innovations $\epsilon_t$ are normally distributed is very restrictive and limited when modeling financial returns as they often exhibit thicker tails and pronounced skenweness that those allowed by the normal distribution. To account for this fact, it is possible to choose a different distribution for the $\epsilon_t$, such as Student's t-distribution or GED[5] distribution that are heavy-tailed and offer more flexibility than the normal when fitting the conditional variance of the time series, this flexibility however, comes at the cost of estimating additional parameters $\boldsymbol{\eta}$ with maximum likelihood, for example the shape or skewness parameters, therefore requiring more computational complexity and resulting in more degrees of freedom. Moreover, each known distribution has a different PDF function so the form of the likelihood function will vary depending on the distribution one has opted to model the innovations with.

### 2.3.1 Forecasting

After the parameters of the model have been estimated with the maximum likelihood approach, they are used to forecast the next values of the time series from the last available observation into the future, h-steps ahead, exploiting the structure of the model and the linear dependence of $y_t$ on the lagged values of the process. The principle underlying time series forecasting is to reduce the expected deviation between the predicted and the actual value of the observations and this is accomplished by minimizing a function of both the predicted and observed values, for example the mean squared deviation, with respect to a vector of parameters to which the foretasted observations depend on. In the maximum likelihood context, recalling (2.95), the negative log-likelihood is minimized by founding the optimal parameters that minimize the conditional squared innovations that represent the squared deviation of $y_t$ from its expected value $\hat{y}_t$. Once the optimal parameters $\hat{\boldsymbol{\omega}}$ are obtained, the best linear prediction h-steps ahead is obtained by computing the expected value of $y_{t+h}$ by conditioning on the information set $F_t$ available at the forecast origin. For a stationary ARMA(P,Q)-GARCH(p,q) of the form:

$$y_t = c + \phi_1 y_{t-1} + \cdots + +\phi_P y_{t-P} + a_t + \theta_1 a_{t-1} + \cdots + \theta_Q a_{t-Q} \tag{2.97}$$

the 1 step ahead observation is given by:

$$y_{t+1} = \hat{c} + \hat{\phi}_1 y_t + \cdots + \hat{\phi}_P y_{t-P+1} + a_{t+1} + \hat{\theta}_1 a_t + \cdots + \hat{\theta}_Q a_{t-Q+1} \tag{2.98}$$

By taking the expectation of (2.98), the 1 step ahead forecast is:

$$\hat{y}_{t+1|t} = \mathrm{E}[y_{t+1|t}] = \hat{c} + \hat{\phi}_1 y_t + \cdots + \hat{\phi}_P y_{t-P+1} + \hat{\theta}_1 a_t + \cdots + \hat{\theta}_Q a_{t-Q+1} \tag{2.99}$$

where $\mathrm{E}[a_{t+1} \mid t] = 0$, by continuing iterating forward, the $2 \leq Q$ steps ahead forecast is:

$$\hat{y}_{t+2|t} = \hat{c} + \hat{\phi}_1 \hat{y}_{t+1} + \sum_{i=2}^{P} \hat{\phi}_i y_{t-i+2} + \sum_{j=2}^{Q} \hat{\theta}_j a_{t-j+2} \tag{2.100}$$

In general, the h-steps ahead forecast is given by:

$$\hat{y}_{t+h|t} = \hat{c} + \sum_{i=1}^{P} \hat{\phi}_i \hat{y}_{t-i+h} + \sum_{j=1}^{Q} \hat{\theta}_j a_{t-j+h} \tag{2.101}$$

[5]Generalized Error Distribution

where $\hat{y}_{t-i+h} = y_{t-i+h} =$ when $(-i+h) \leq 0$ and $a_{t-j+h} = 0$ if $(-j+h) > 0$. Moreover, because for an ARMA process the correlation between the shock terms will die out as soon as the lag $|h| > Q$, the forecasts will depend only on the autoregressive part such that:

$$\hat{y}_{t+h|t} = \hat{c} + \sum_{i=1}^{P} \hat{\phi}_i \hat{y}_{t-i+h} \tag{2.102}$$

for $h = \{Q+1, Q+2, \dots\}$ and for $h \to \infty$, given that the process is stationary such that $|\phi| < 1$, by recursive substitution of $\hat{y}_{t-i+h}$ in (2.102) the coefficient of the forecasts in the summation will become $\hat{\phi}_i^h$ and will decay geometrically to zero demonstrating that the forecasts for a number of steps forward in time will converge to the unconditional mean of the process.

As regards the forecast error, $e(h) = y_{t+h} - \hat{y}_{t+h|t}$, it will increase with h. Indeed, by expressing the process in its MA($\infty$) form:

$$y_t = \mu + \sum_{j=0}^{\infty} \Psi_j a_{t-j} \tag{2.103}$$

$$\hat{y}_{t+h|t} = \hat{\mu} + \sum_{j=h}^{\infty} \hat{\Psi}_j a_{t-j+h} \tag{2.104}$$

the error and the variance of the error are:

$$e(h) = y_{t+h} - \hat{y}_{t+h|t} = \sum_{j=0}^{h-1} \hat{\Psi}_j a_{t-j+h} \tag{2.105}$$

$$Var[e(h)] = \sum_{j=0}^{h-1} \hat{\Psi}_j^2 \, \sigma_{t-j+h}^2 \tag{2.106}$$

such that, the variance increases when $h$ increases and as $h \to \infty$ it converges to the variance of the infinite process (2.103):

$$Var[e(h)] = \sigma_a^2 \sum_{j=0}^{\infty} \hat{\Psi}_j^2 \tag{2.107}$$

where $\hat{\Psi}_0^2 = 1$ and $\sigma_a^2 = \frac{\alpha_0}{1-\sum_{i=1}^{max(p,q)}(\alpha_i+\beta_i)}$ is the unconditional variance of the conditional heteroscedastic part and this demonstrates that the convergence to the long-run level is true for GARCH(p,q) too, provided that the parameters respect the positivity constraints.

## 2.4 Data Analysis

Before fitting the models, a sound approach would be to check if the time series needs a preliminary transformation in order to coherently meet the hypothesis of the theoretical models. Bitcoin is traded every day of the week for every month of the year, its market is highly speculative so the price evolution is highly volatile and like most financial speculative assets, exhibits periods of mean reversion alternated with period of very persistent momentum characterized by a positive, or negative, large returns for multiple trading days in a row, so it hardly can be classified as a stationary process and a proper transformation of the series should be performed. Throughout the entire sections I will perform all the calculations with R as it is a very practical and powerful software providing plenty of packages to carry out statistical data analysis, time series modeling and parameters estimation. The data set in analysis comprises the historical 4 hour close prices of BTC/USDT[6] from 2017-08-17 to 2020-11-30 resulting in 7194 observations. To have a glance of the probability density of the sample in question, in figure 2.1(a) below, I have plotted an histogram of the close prices of BTC/USDT along with the fitted density of a normal distribution, represented by the dashed red line, with mean equal to the sample median and with standard deviation equal to the MAD of the sample. I have chosen the median and mean absolute deviation from the median because the MAD is more robust with respect to outliers and provides a less biased approximation for the dispersion of the sample. The plot shows a substantial difference between the two densities and BTC/USDT exhibits a pronunced right skewness, the outliers make the data more dispersed from the center of the distribution and the thicker tails are all symptoms of departure from a normal distribution. The non-normality of the sample is purported by the Quantile-Quantile plot in figure 2.1(b), where the red line interpolates the theoretical first and third quartiles of a normal distribution, showing a clear non-linear relationship between the sample, with its quartiles plotted on the x-axis, and the normal distribution. The concave shape in the bottom left of the plot confirms the precence of positive skewness in the data, and around the third quartile the plot becomes even more complex hinting that the data is showing a multi-modal behaviour and a simple normal distribution is far from a good fit.

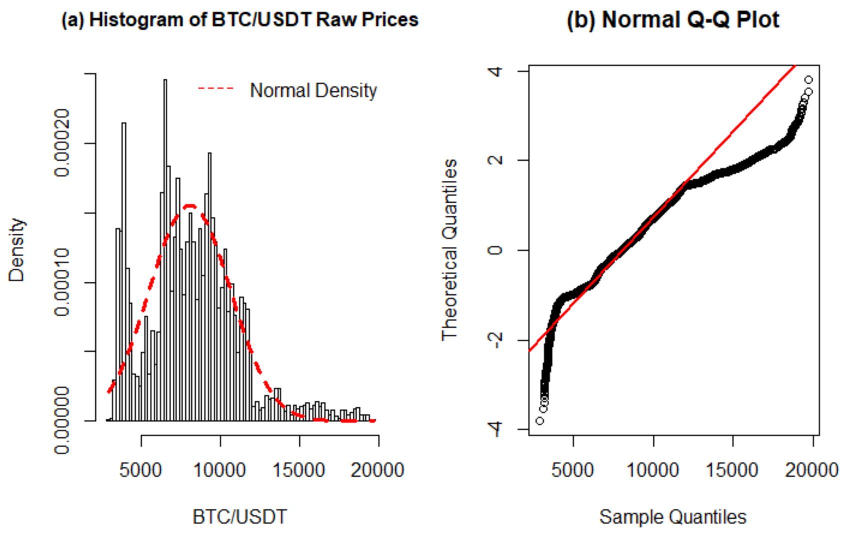


Figure 2.1: Analysis of BTC/USDT 4h close

[6]USDT is the stablecoin TheterUSD which is pegged against the US dollar. I am using this pair because on the Binance exchange there is no possibility to purchase BTC directly with USD, or other fiat currencies, and one has to first convert fiat to a stablecoin or an other cryptocurrency in order to start trading BTC.

To scale down and standardize the data, I have computed the returns of BTC/USDT and made the same plots as before, the results are appreciable in figure 2.2. Now the returns seems more evenly distributed around the center of the data but the convex-concave shape of the Q-Q plot indicates heavier tails than the theoretical normal distribution and corroborating the assumption of non-normality of returns, the Jarque-Bera test [25], which jointly tests the departure of the sample skweness and kurtosis from the corresponding 0 and 3 for a normal distribution, and the Shapiro-Wilk [26] test, based on the correlation between the expected order statistics of the standard normal distribution and the sample ones, both yielded a p-value of 2.2e-16 going against the null hypothesis of normality.

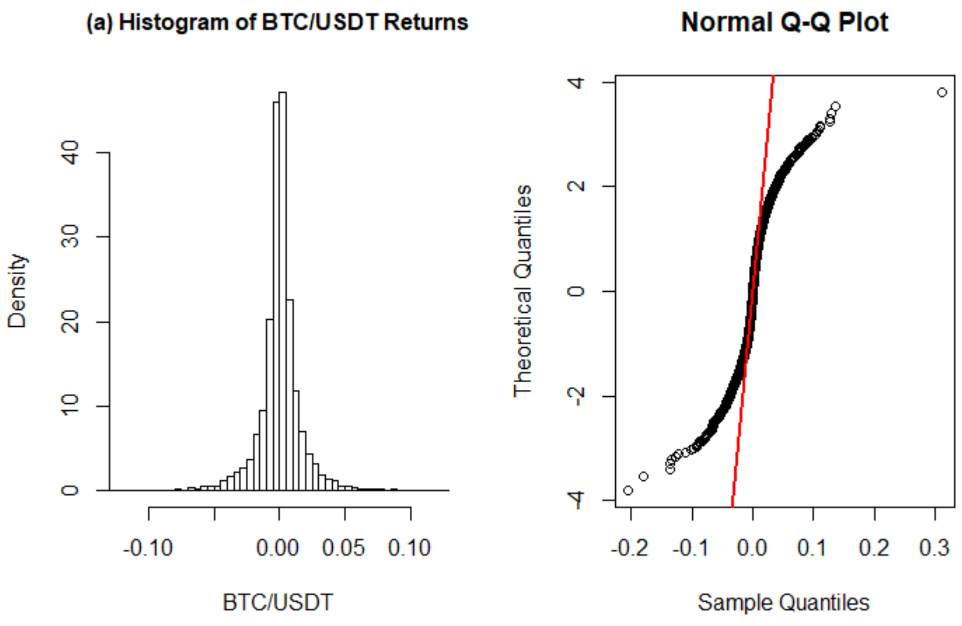


Figure 2.2: Analysis of BTC/USDT returns

In order to stabilize the variance and induce some symmetry in the data I have applied a log transformation to the time series. Moreover, the asset pricing theory [27] supports working with log-returns which have better statistical properties and are arithmetically easier to handle compared to raw returns. Indeed, if $P_t$ and $P_{t-1}$ are the price of an asset at time t an t-1 respectively, by taking the logarithm of prices such that $p_t = log(P_t)$ the return is simpler to compute because of the properties of logarithms, as a matter of fact, if the gross return is computed as $1 + R_t = \frac{P_t}{P_{t-1}}$ the 1 period log return is given by $r_t = log(1 + R_t) = p_t - p_{t-1}$ and for moderately small values of $R_t$, $log(1 + R_t) \approx R_t$. Thanks to this property, the calculation of multi-period compounded return is more practical with log-prices because it involves the sum of the log-returns and not the product of gross returns anymore. Additionally, the theory states that if the log-returns are i.i.d. normal, the log prices follow a lognormal geometric random walk, so to explore this hypothesis I have reported the Q-Q plots for log prices and log returns. In practice, the assumption of i.i.d. log returns is too restrictive and as can be noticed from figure 2.3, the non-normality of the logged time-series still persists, the log-returns exhibit heavier tails than a normal distribution, so the hypothesis of the lognormal geometric random walk does not hold in this case, even though the log transformation actually reduced the asymmetry of the data as the skweness of the sample lowered, in absolute value, from 0.43 for the returns to -0.23 for the log-returns.

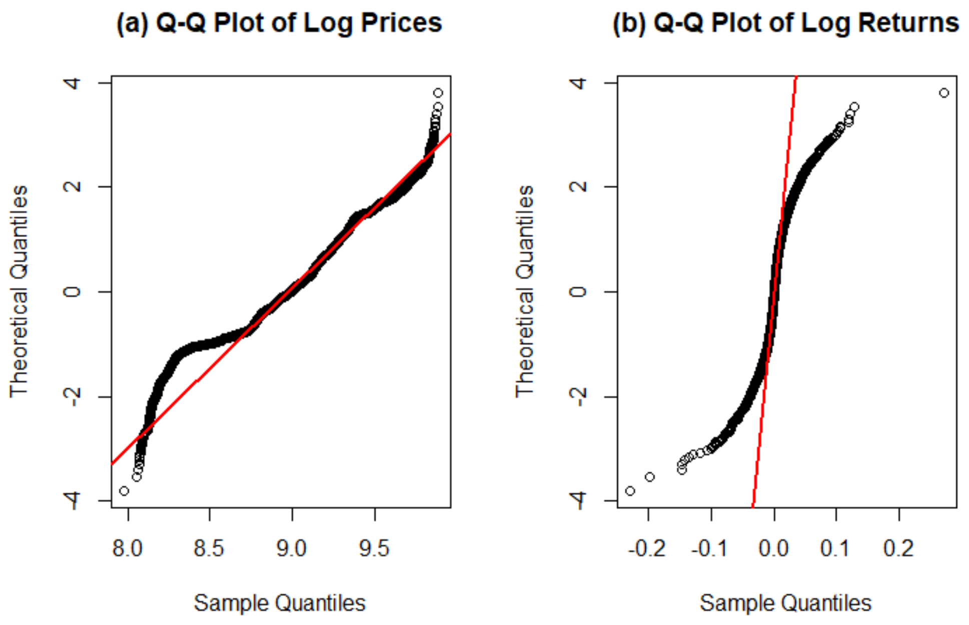


Figure 2.3: Log transformation

By retaining the log transformed series, applying the difference operator to the log-prices makes more sense than differencing raw prices as, for example, if the logged series is found to be an integrated process of order one, $I(1)$, the first difference will result in log-returns and one can exploit the properties of logarithms. In order to identify the nature of the logged time series and test if it is a stationary process, I plotted the log-prices against the time index and from figure 2.4(a) the process seems like wiggling around in some time intervals and wondering without reverting to a single, specific level, exhibiting a behaviour similar to a random-walk, so it might be considered as an $I(1)$ and, by taking a first order difference, I obtained the figure 2.4(b) which shows mean-reversion around zero and supports the hypothesis that the log-price is a random walk whose difference has a demeaning effect and deleted a possible linear deterministic trend. Hence, the log returns might possibly be considered as a stationary process and to check the stationairty assumption I run three different test for stationarity, the augmented Dickey-Fuller [28] test and the Phillips-Perron [29] test, which both check the presence of unit root test by fitting an AR(p) model to the argument with the null hypothesis being unit root, and the KPSS [30] test which instead checks if the null hypothesis of stationarity holds for the selected sample. The output of the tests are below and all support the hypothesis of stationarity for the log-returns with a very satisfactory p-value so the transformed series should not have explosive moments and could probably be fitted by stationary models.

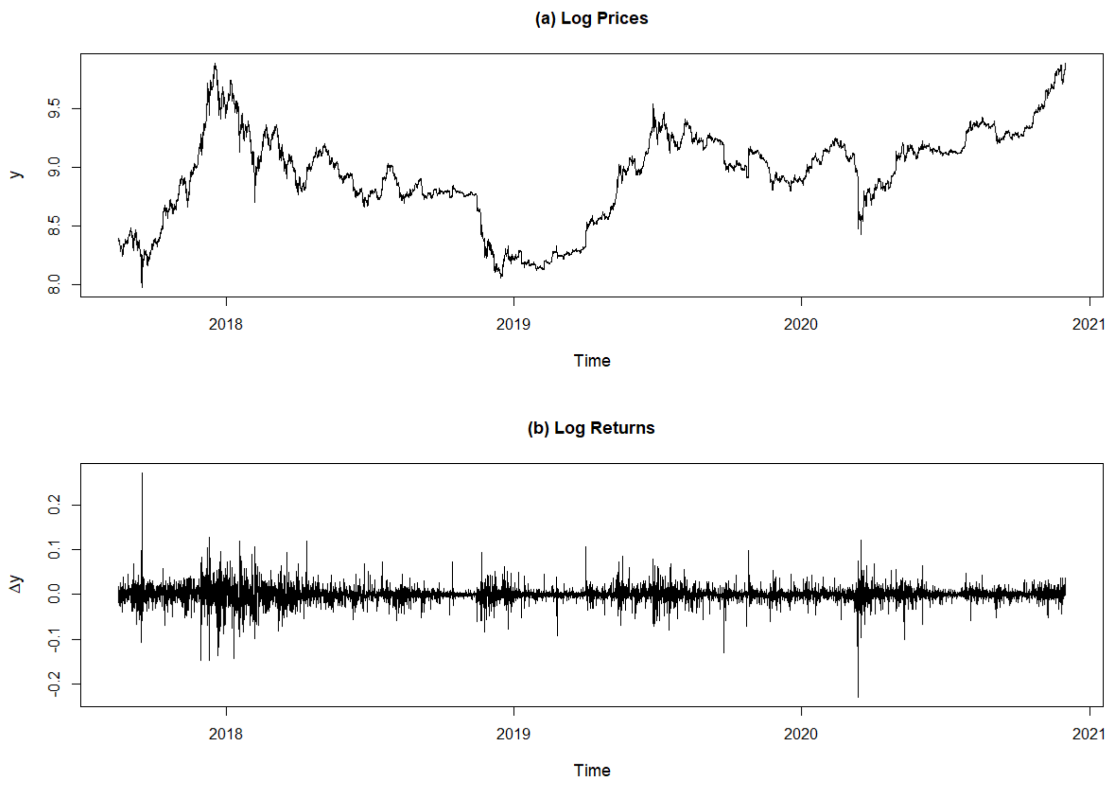


Figure 2.4: Integrated process and first difference

```
        Augmented Dickey-Fuller Test
data:  log.ret
Dickey-Fuller = -18.646, Lag order = 19, p-value = 0.01
alternative hypothesis: stationary
p-value smaller than printed p-value

        Phillips-Perron Unit Root Test
data:  log.ret
Dickey-Fuller Z(alpha) = -7798.5, Truncation lag parameter = 11, p-value = 0.01
alternative hypothesis: stationary
p-value smaller than printed p-value

        KPSS Test for Level Stationarity
data:  log.ret
KPSS Level = 0.11884, Truncation lag parameter = 11, p-value = 0.1
p-value greater than printed p-value
```

The assumption of stationarity is purported by the ACF plots too, indeed, in figure 2.5 one can see from the plot on the left a strong, lasting autocorrelation of the log-prices over time, even at further lags, symptom of non-finite moments and of non-stationarity. After taking the difference of the series, now the ACF is less pronounced and while for distant lags, from the $10th$ and beyond, the ACF spikes fall within the test bound, which by default are at 0.05 level, or at least they are well contained around that level, the ACF at short term lags indicates some type of minor, less strong dependence that might be explained by a time series model and might recall the behavior of one of the process explored in the previous sections. Therefore, the ACF plots provide a substantial basis for model and order identification and I will rely on them, although not blindly, but accompanying them with other indicators and statistics that will help me decide which model is the most appropriate and fits the best.

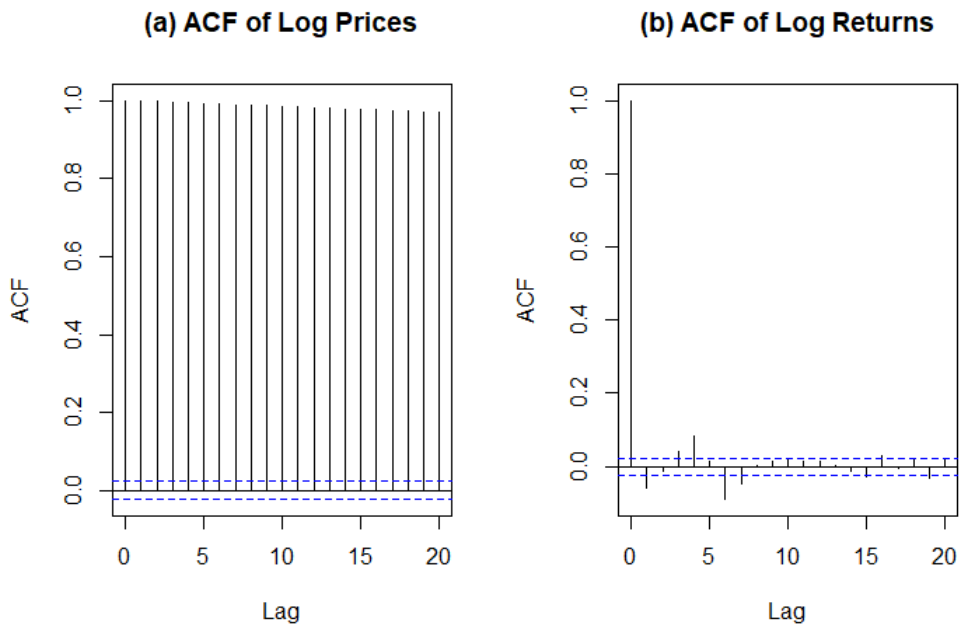


Figure 2.5: Autocorrelation function of the transformed series

### 2.4.1 Model Selection Criteria

Selecting a good model encompasses optimizing the so called bias-variance trade off, that is, choosing with parsimony a model that helps explaining the behaviour and the dependencies between the data in the sample while still providing a good performance out-of-sample when forecasting. The risk of employing a model with too much parameters is overfitting the data, namely, while it might provide an adequate performance when inferencing about the distribution of the sample, thus reducing the bias, it might lead to a rather complex, thus difficult to interpret, estimator degrading the out-of-sample performance, increasing the variance, and consequently leading to an inaccurate forecast of the time series. On the other side, choosing a model with too few parameters, in order to reduce the variance of the estimator, might lead in underfitting the in-sample data and increase the bias due to a low explanatory power. To asses the goodness of fit while still accounting for model complexity, I will rely on two information criteria widely used in practice, namely the Akaike's [31] and Bayesian information criterion [32], abbreviated AIC and BIC respectively, defined as:

$$AIC = -2\ell(\hat{\boldsymbol{\omega}}; y_N) + 2p \tag{2.108}$$

$$BIC = -2\ell(\hat{\boldsymbol{\omega}}; y_N) + log(N)p \tag{2.109}$$

Both of them are functions of the log-likelihood such that, when choosing which model to retain among the candidates, the ones with the highest log-likelihood, therefore with the lowest AIC and BIC values, provide the best approximation of the underlying process and achieve an optimal bias-variance trade off. The advantage of using AIC and BIC when comparing different models is that they account for model complexity, indeed the rightmost factors in (2.108) and (2.109) penalize the models involving too much parameters,$p$, to obtain a satisfactory log-likelihood value. Between the two, the BIC is the most restrictive one because it factors in the penalty term the size of the sample, rewarding the simpler models and sometimes might lead to a different conclusion than AIC, nonetheless both are minimized by the most parsimonious models and they are excellent model selection criteria.

### 2.4.2 Model Identification

After checking that logging and differencing the series provides a satisfactory transformation and the log returns appears stationary, now the plot of the sample ACF and PACF can help identifying if the series can be fitted by an AR or MA process and what should be their order in case they are employed. Form figure 2.6(a) the ACF at lag one rapidly decreases almost to zero showing the typical behaviour of a moving average process, and after that, from lag 3 the autocorrelation alternates signs and slowly decays to zero at lag 8, this could probably the effect of an autoregressive part with coefficients high in magnitude and with opposite signs. Coherently with these findings, the PACF in figure 2.6(b) shows even more clearly the characteristic of the PACF of an AR process which abruptly decreases after the first lag, dropping to more than half and on subsequent lags, the PACF of the process exhibits a gradual decay that supports the hypothesis of the presence of an additional MA part.

To fit an initial ARIMA(P,d,Q) to the log prices, I have used the R function *arima()* which estimates the parameters via maximum likelihood using conditional least-squares estimates as starting values for the recursion. I run a loop to jointly estimate the parameters of the model over a grid of five values per order, $P = \{0, \ldots, 4\}$ and $Q = \{0, \ldots, 4\}$ with $d$ fixed to 1, resulting in 25 iterations, with the AIC and BIC relative to each model. I also tested separately if an higher order

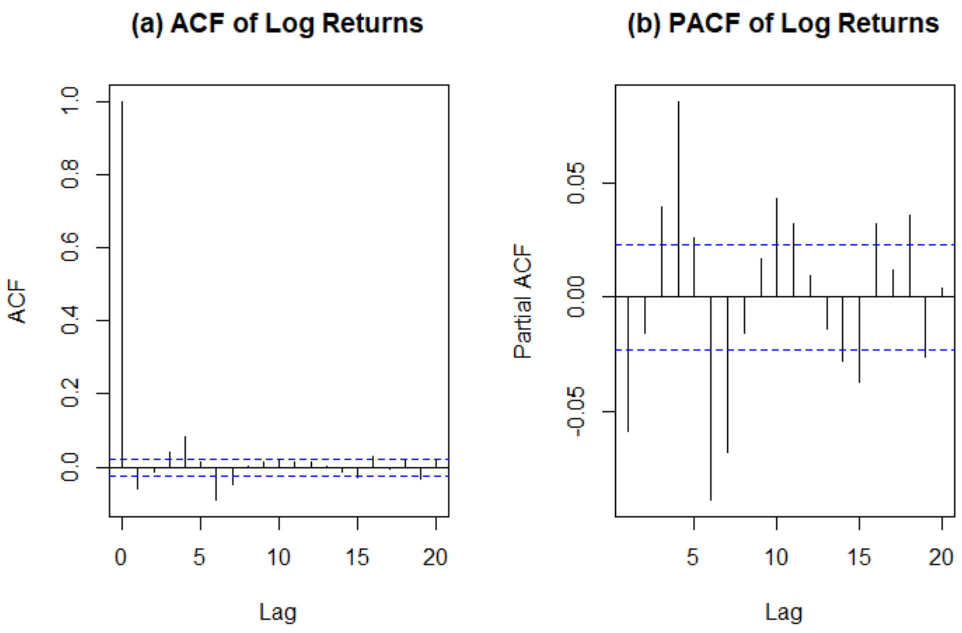


Figure 2.6: Autocorrelation and partial autocorrelation functions of log returns

of differencing was needed but the outcomes always reported that $d = 1$ was enough and higher values introduced a bad overdifferecing that reduced both AIC and BIC and inflated the standard errors of the parameters. Additionally, I opted to remove the mean from the set of parameters to estimate because from the time plot of log returns they seemed to oscillate around a mean level equal to zero and when including $\mu$ in the model, its estimate was always statistically not different from zero and was never beneficial in terms of AIC and BIC.

The plot in figure 2.7 reports the absolute values of AIC and BIC for each iteration, showing that both criteria reached a local maximum on iteration 14 and 13 respectively, and after that, the AIC started diverging while the BIC remained stable, due to its higher penalty, without further improvements, signaling that an optimal set of parameters that minimized the criteria was already found at an earlier iteration.

Table 2.1 reports the effective values of AIC and BIC for some of the iterations, along with the relative orders of the model. The values in bold indicate that the best values for AIC and BIC were achieved, respectively, by an ARIMA(2,1,3) and ARIMA(2,1,2) with mean zero and the results are coherent with the analysis of the ACF and PACF made before. Even if BIC choose the $P = 2, Q = 2$, its value is very similar in the following iteration and coherent with the AIC obtained on the same iteration, moreover, when looking at the estimated parameters of ARIMA(2,1,2), the first moving average parameter was over one in magnitude, so to avoid any instability problems, I was more inclined to lean towards the other model.

The output of the ARIMA(2,1,3) parameter estimation is found below and the high magni-

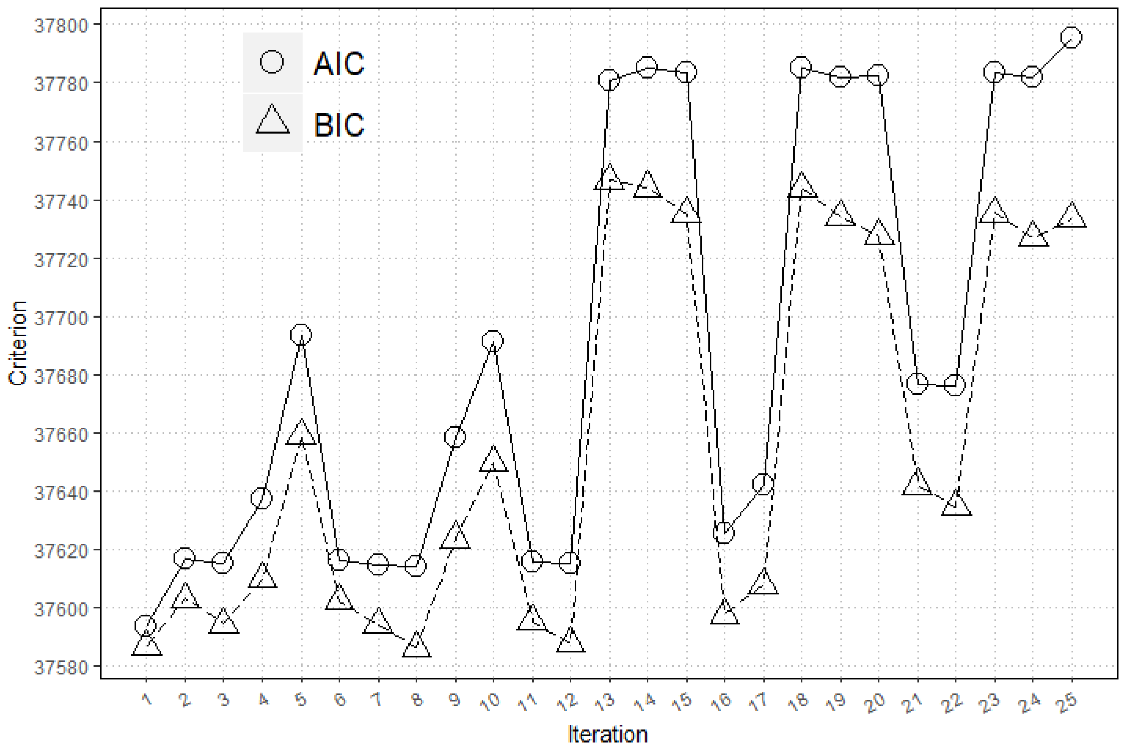


Figure 2.7: Absolute values of AIC and BIC for every iteration

| P | Q | AIC | BIC |
|---|---|---|---|
| 2 | 1 | -37615.09 | -37587.57 |
| 2 | 2 | -37781.03 | **-37746.63** |
| 2 | 3 | **-37785.30** | -37744.02 |
| 2 | 4 | -37783.31 | -37735.14 |
| 3 | 0 | -37625.20 | -37597.68 |
| 3 | 1 | -37642.31 | -37607.91 |
| 3 | 2 | -37785.23 | -37743.95 |
| 3 | 3 | -37782.02 | -37733.86 |
| 3 | 4 | -37782.47 | -37727.43 |

Table 2.1: AIC and BIC Values for Iterations 12 to 20

tude of the autoregressive parameters, expecially of the $\hat{\phi}_1$, might raise some concerns about the stationarity of the process, therefore I checked the root of the AR polynomial using the *polyroot()* function and found that the absolute values of the root were safely outside the unit circle[7]. Besides that, the first four parameters are well above their standard errors and statistically significant, whereas $\hat{\theta}_3$ has a low value and is about 2.5 times over its s.e., resulting in a p-value $\approx 0.01094$

[7]The polynomial $1 - 0.9153x + 0.7051x^2 = 0$ has complex roots and their absolute value measures the distance between the origin of the complex plane and $z$.

which is still significant but only at 0.05 level.

```
Call:
arima(x = log.prices, order = c(2, 1, 3), include.mean = FALSE)

Coefficients:
         ar1      ar2      ma1     ma2     ma3
      0.9153  -0.7051  -0.9881  0.7732  0.0389
s.e.  0.0370   0.0306   0.0385  0.0362  0.0153

sigma^2 estimated as 0.0003058:  log likelihood = 18898.65,  aic = -37785.3
```

### 2.4.3 Model Diagnostic

If ARIMA(2,1,3) fits the series well, the residuals should look like white noise and show no autocorrelation, hence to check if this assumptions holds, I plotted the residuals of the model against time, their ACF and a normal Q-Q plot. While on figure 2.8(a) the residuals look like oscillating around zero, they exhibit a consistent volatility clustering and heavy tails when compared to a normal distribution, so employing a conditional variance model like GARCH might explain this behaviour more appropriately. The short-term ACF is well inside the confidence intervals but there is a small spike at lag 6 that raises some concerns about the validity of the white noise assumption for the residuals.

Running the Ljung-Box [33] test, which tests if the autocorrelations up to H lags are simultaneously equal to zero, the p-value with H=12 was 0.0001399 for the ARIMA(2,1,3) and even lower for ARIMA(2,1,2), rejecting the null hypothesis that $\rho(1) = \rho(2) = \cdots = \rho(H)$ in favour of the alternative that at least one is non zero, suggesting that the hypothesis of white noise is violated. This result indicates that the log returns have a persistent long-memory and might be better modeled by long-memory processes by employing fractional differencing. However, even if the ACF is statistically different from zero, the spike at lag six is not very large and far from the bounds so in terms of practical significance the results form the previous test might not totally undermine the usefulness of the model when forecasting. Lastly, the residual autocorrelation could be due to random variation, so I performed the Durbin-Watson [34] test which checks if the null hypothesis of no residual correlation holds by running N simulations, and the results with N=10,000 and lag=10 were all corroborating $H_0$ as the value of the statistic, whose range goes from 0 (positive autocorrelation) through 2 (no autocorrelation) to 4 (negative autocorrelation), was around 2 at all lags and 2.1 at lag six.

### 2.4.4 Modeling Time-Varying Volatility

An analysis of the autocorrelation of squared residuals of the fitted model can be useful to test if there are ARCH effects in the squared residuals and asses if the assumption of constant volatility

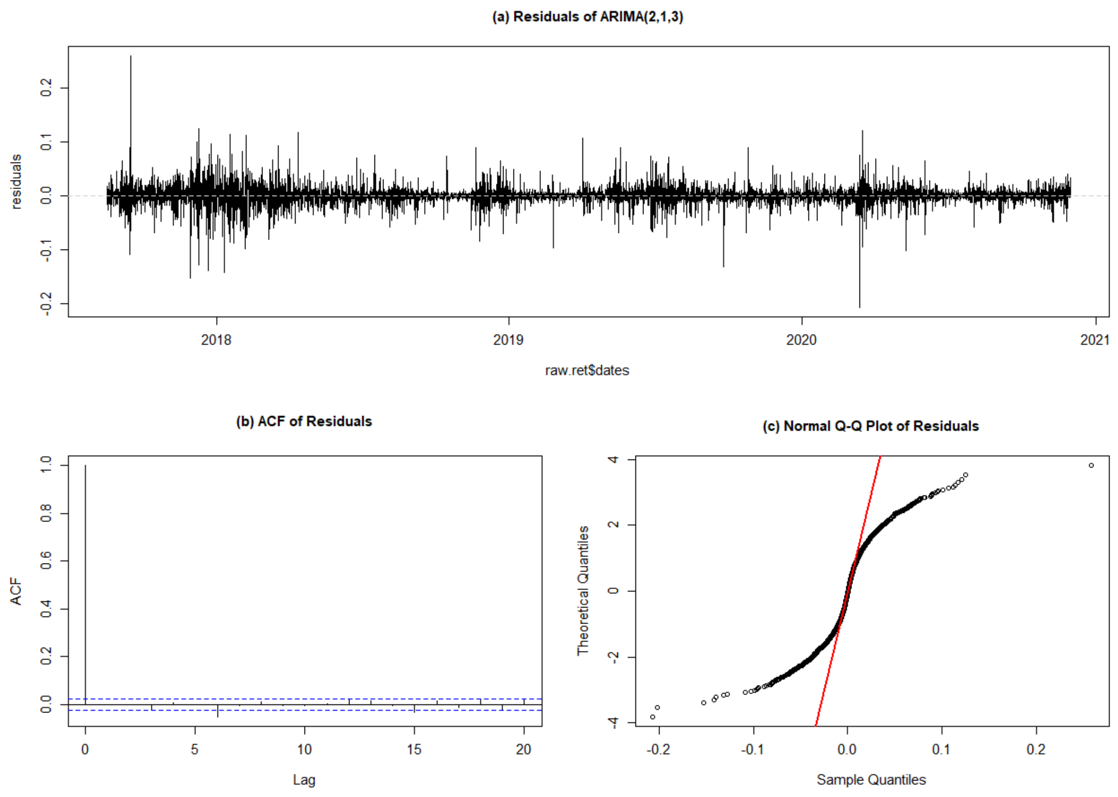


Figure 2.8: Analysis of residuals

in the conditional mean model can affect forecasts. Indeed, the behaviour of $\rho_{a^2}(h)$ in Figure 2.9 indicates the presence of conditional heteroscedasticity, as the squared log-returns would do, which correlates to the volatility clustering seen during the most bustling trading days, as the ACF in this case is decaying rather slowly, persisting across all displayed lags, and hardly respects the confidence intervals.

By allowing the constant innovation process to follow an autoregressive process instead, I tested if an ARCH(p) or GARCH(p,q) model would fit the conditional standard deviation of the series. For this purpose, with the same approach used before, I searched over a grid of orders $p$ and $q$ which combination would yield the best performance in terms of AIC and BIC, and checked which theoretical distribution is better suited to approximate the empirical distribution of the standardized residuals $\epsilon_t = \frac{a_t}{\sigma_t}$. Employing the *rugarch* package, I started by fixing $q = 0$ and $p = 1, 2, \dots$ in order to see if only ARCH(p) was enough to explain the autocorrelation in the squared residuals and, assuming i.i.d. normally distributed $\epsilon_t$, the model that minimized both information criteria was ARIMA(2,1,3)-ARCH(12) with AIC=-34712 and BIC=-34617 resulting in unsatisfactory results in terms of statistic significance of its parameters over the fifth lag, the over-parametrization led to an high variance out-of-sample dampening the root mean squared error between the actual and forecasted values of log returns, purporting the addition in the conditional variance formula of its lagged values to obtain a more parsimonious model and better

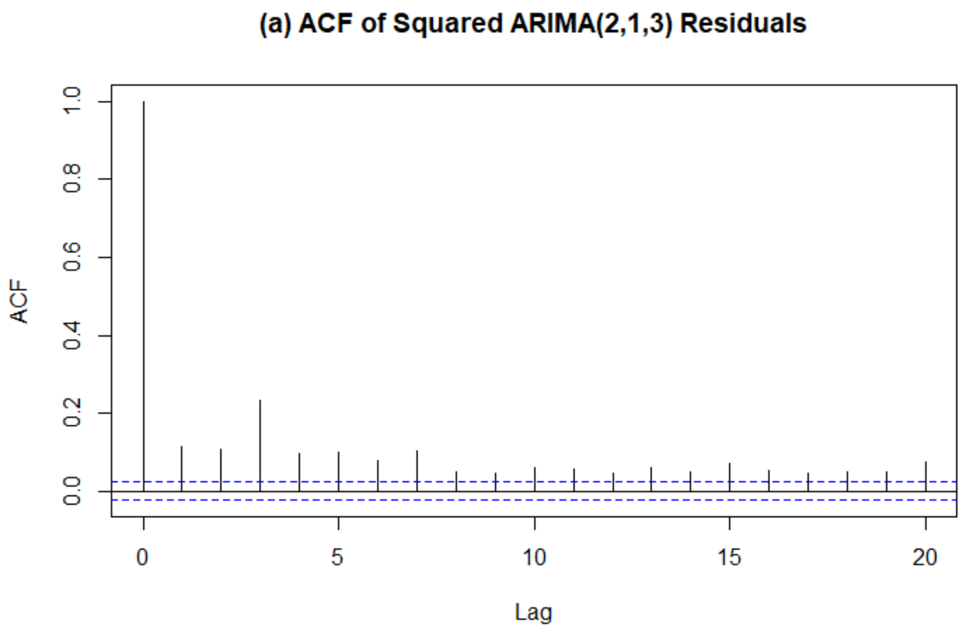


Figure 2.9: ARCH effects

o.o.s. forecasts of both conditional mean and variance.

When allowing $q$ to vary over the grid along with $p$ to introduce GARCH effects in the model, and considering both ARIMA(2,1,3) and ARIMA(2,1,2) for the conditional mean, with the same assumption on standardized residuals as before, the optimal model was found to be ARIMA(2,1,3)-GARCH(1,5) according to AIC, with a value of -34841, and ARIMA(2,1,3)-GARCH(1,4) for BIC, which attained a local minimum at -34774. Both specifications show that accounting for the lagged values of $\sigma_t^2$ can be beneficial for the goodness of fit and the GARCH(p,q) can be used to approximate the persistence of the conditional variance better than ARCH(p) alone, even though, by looking at the results[8] for the BIC selected model, the parameters of the autoregressive moving average part are very similar, signaling a possible parameter redundancy. The GARCH parameters are all significant and while the weighted Ljung-Box test on standardized residuals is still indicating residual autocorrelation after the first lag, the same test on standardized squared residuals are more in favour of the the null hypothesis of absence of serial correlation along with the Weighted ARCH LM Tests.

```
GARCH Model      : sGARCH(1,4)
Mean Model       : ARFIMA(2,0,3)
Distribution     : norm

Optimal Parameters
------------------------------------
```

[8]*rugarch* normalizes the values of AIC and BIC in the output of the fitting routine so their values have to be multiplied by 6146 to get the effective ones.

```
        Estimate  Std. Error  t value Pr(>|t|)
ar1     0.933222    0.036697  25.4307 0.000000
ar2    -0.774713    0.037449 -20.6869 0.000000
ma1    -0.951680    0.039314 -24.2072 0.000000
ma2     0.793323    0.041784  18.9865 0.000000
ma3     0.048532    0.016870   2.8768 0.004018
omega   0.000009    0.000001  13.7320 0.000000
alpha1  0.216805    0.007416  29.2338 0.000000
beta1   0.181206    0.039720   4.5621 0.000005
beta2   0.185659    0.042377   4.3811 0.000012
beta3   0.211441    0.046735   4.5243 0.000006
beta4   0.195984    0.034195   5.7314 0.000000

LogLikelihood : 20398

Information Criteria
------------------------------------
Akaike       -5.6686
Bayes        -5.6580

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic  p-value
Lag[1]                       2.896 0.088779
Lag[2*(p+q)+(p+q)-1][14]    15.809 0.000000
Lag[4*(p+q)+(p+q)-1][24]    23.895 0.000238
d.o.f=5
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
Lag[1]                      0.7089  0.3998
Lag[2*(p+q)+(p+q)-1][14]    3.9268  0.8871
Lag[4*(p+q)+(p+q)-1][24]    6.5761  0.9498
d.o.f=5

Weighted ARCH LM Tests
------------------------------------
             Statistic Shape Scale P-Value
ARCH Lag[6]     0.2287 0.500 2.000  0.6325
ARCH Lag[8]     1.4674 1.480 1.774  0.6397
ARCH Lag[10]    3.5760 2.424 1.650  0.4815

Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
```

```
1    20      1347    3.144e-274
2    30      1435    3.442e-284
3    40      1456    2.380e-280
4    50      1501    4.104e-282
```

The adjusted Pearson [35] test, however, reports a very low p-value leading to reject the null hypothesis that $\epsilon_t$ follows a Gaussian distribution due to a low goodness of fit. The same concern stems up from the Q-Q plot below which shows that the distribution of the standardized residuals is heavier on the tails than a normal. Moreover, the empirical density in the left plot of figure 2.10 has heavy peaks around the sample median, corroborating the fact that a normal distribution is too strict of an assumption, leading me to try different distributions for the standardized residuals, namely, the Student's t-distribution and GED. I fitted both t- and GED distributions to $\epsilon_t$ to assess

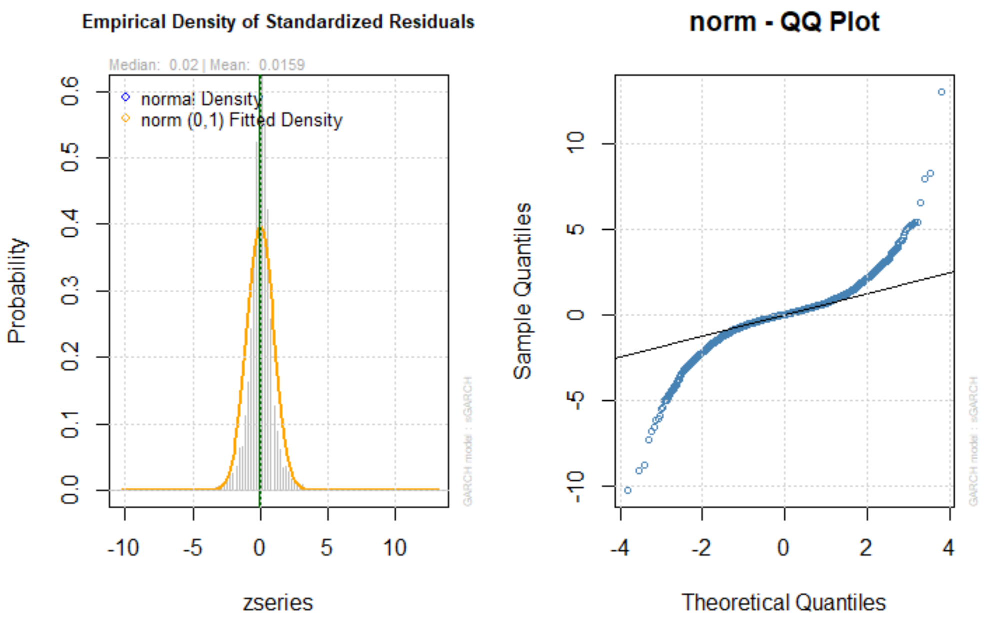


Figure 2.10: Analysis of residuals of ARIMA(2,1,3)-GARCH(1,4)

how much they can improve the goodness of fit and approximate the heavy tails of the standardized residuals. The maximum likelihood estimates for the shape parameter of the fitted t- was $\eta = 2.38$, and while confirming the pronounced tail thickness of the residuals, this results rises some doubts about employing the t- because for $\eta \leq 4$ its kurtosis is not defined, whereas, the mle's for the fitted GED were $\eta = 0.84$ for the shape and 0.95 for the standard deviation. Considering that GED is very flexible when fitting an unknown distribution as it generalizes different distributions depending on the parameter values, the latter results makes sense because when $\eta = 1$ the GED is equal to a Laplace distribution which is characterized by a sharp and elongated shape around the mean, recalling the shape of the empirical distribution seen before.

The Q-Q plots make even more clear that either t- or GED can improve the goodness of fit and

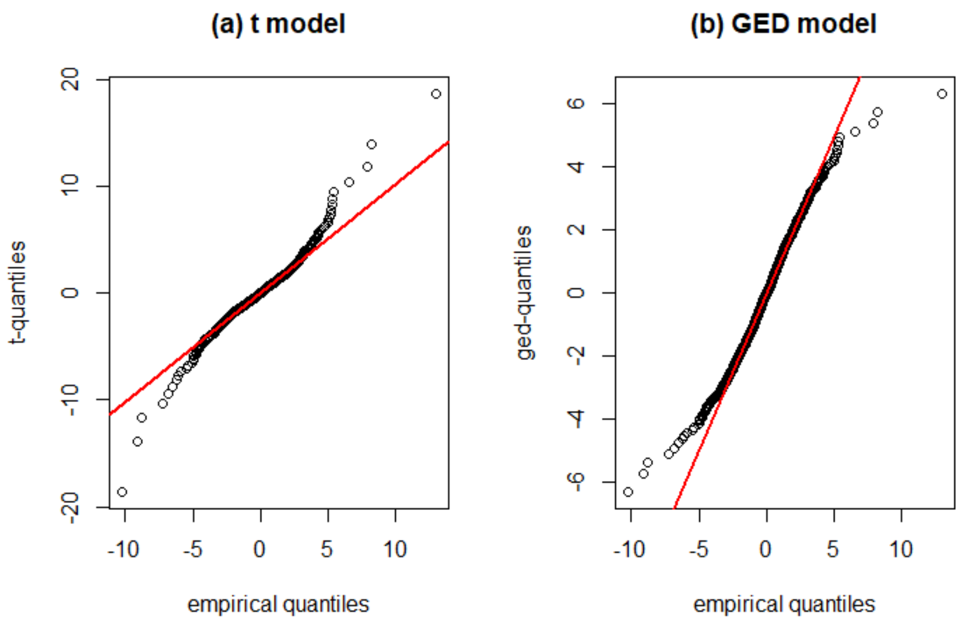


Figure 2.11: Q-Q plots of the residuals against theoretical quantiles

be a better alternative than the Gaussian distribution when fitting the conditional heteroscedasticity. By an iterative approach, I tested for each of the two conditional mean models specified earlier the various combinations of the orders $p$, $q$ of the GARCH part and the results confirmed that the polynomial tails of t- or the double-exponential ones of GED are better suited in this case and improved by a consistent margin the goodness of fit when compared to the model under Gaussian $\epsilon_t$ assumption. Both AIC and BIC decreased and consistently chose parsimonious models with fewer parameters than before, signaling that the two heavy tailed distributions approximate better the behaviour of the series. This time though, I did not pick a model based solely on the information criteria but also by looking at the performance of each model out of sample, the values and statistical significance of the parameters and the results of the test statistics. Table 2.2 is a summary of the model specifications that minimized the information criteria and performed the best during each iteration, and as one can see, BIC tended to pick models with half the order of $q$ compared to AIC, thus resulting in more parsimonious choices. Even if the models with t- distributed residuals seem to perform the best in terms of both criteria when compared to the GED counterpart, for all of them the parameters of the GARCH part were statistically not different from zero, the AR and MA terms were higher than one in magnitude, which could be due to the non-definiteness of the t's kurtosis as the estimated $\eta$ was pretty consistent with the one obtained before, and both the out of sample performance metrics are worse then the other specifications. Ohterwise, when imposing a GED distribution to the standardized residuals, the performance in therms of RMSE and MAE was found to be better when forecasting returns out of sample and

I was lead to choose the ARIMA(2,1,2)-GARCH(1,2) because its BIC and AIC are very close to the other's and, among the specifications in this series of iterations, was the model with the most consistent and significant parameters and with an acceptable RMSE and MAE. This results shows that in this case, employing GED helps improving the g-o-f to the extent that now the parameters of ARIMA(2,1,2) part, which presented some problems before, are now more stable in terms of magnitude and now is preferred over ARIMA(2,1,3) for modeling the conditional mean, and with the GARCH extension offers the best bias-variance trade-off.

| | P | Q | p | q | AIC | BIC | RMSE | MAE |
|---|---|---|---|---|---|---|---|---|
| t | 2 | 2 | 1 | 2 | -37070 | -37017 | 0.0097403 | 0.0077427 |
| | 2 | 2 | 1 | 4 | -37074 | -37010 | 0.0097396 | 0.0077423 |
| | 2 | 3 | 1 | 2 | -37070 | -37011 | 0.0097266 | 0.0077369 |
| | 2 | 3 | 1 | 4 | -37074 | -37004 | 0.0097223 | 0.007734 |
| GED | 2 | 2 | 1 | 2 | -37040 | -36987 | 0.0096966 | 0.0076803 |
| | 2 | 2 | 1 | 4 | -37045 | -36980 | 0.0096963 | 0.0076795 |
| | 2 | 3 | 1 | 2 | -37048 | -36989 | 0.0097256 | 0.0077387 |
| | 2 | 3 | 1 | 4 | -37053 | -36983 | 0.0097235 | 0.007738 |

Table 2.2: Summary of the models

The results of ARIMA(2,1,2)-GARCH(1,2) fitting routine is reported below, the ACF of the squared residuals (firgure 2.12) is inside the confidence bounds for all short-term lags and both the Q-Q and the density plots show that GED is far better that the Gaussian distribution for the series, and even if the Pearson test has a low p-value, this could be driven by the high frequency of the data and the fact that outliers are accounted to be more statistically significant by the tests in large samples like this one.

```
GARCH Model      : sGARCH(1,2)
Mean Model       : ARFIMA(2,0,2)
Distribution     : ged

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.782106    0.006153  127.10832 0.000000
ar2    -0.673898    0.021833  -30.86605 0.000000
ma1    -0.850040    0.005986 -141.99803 0.000000
ma2     0.717206    0.019242   37.27370 0.000000
omega   0.000002    0.000003    0.53369 0.593556
alpha1  0.100313    0.036647    2.73732 0.006194
beta1   0.377777    0.127994    2.95152 0.003162
beta2   0.520909    0.100434    5.18657 0.000000
shape   0.823067    0.020471   40.20671 0.000000
```

```
Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                          statistic   p-value
Lag[1]                        35.05 3.208e-09
Lag[2*(p+q)+(p+q)-1][11]      55.14 0.000e+00
Lag[4*(p+q)+(p+q)-1][19]      66.05 0.000e+00
d.o.f=4
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                          statistic p-value
Lag[1]                       0.2802  0.5966
Lag[2*(p+q)+(p+q)-1][8]      0.6930  0.9905
Lag[4*(p+q)+(p+q)-1][14]     1.9158  0.9927
d.o.f=3

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[4]   0.04911 0.500 2.000  0.8246
ARCH Lag[6]   0.39916 1.461 1.711  0.9196
ARCH Lag[8]   0.93660 2.368 1.583  0.9334

Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     98.22    1.121e-12
2    30    126.12    4.456e-14
3    40    123.31    1.113e-10
4    50    147.45    8.576e-12
```

Figure 2.13 shows the actual log-returns of BTC/USDT over the sample period with the fitted conditional standard deviation superimposed, computed as $\hat{y}_t \pm 1.96\hat{\sigma}_t$, and the grey line is representing the fitted conditional mean of the time series. The latter is driven by the ARIMA part so it is expected to be equal to the unconditional long-run mean of the series, whereas the GARCH part does an great job at fitting the conditional heteroscedasticity and provides a satisfying approximation of the wide scope of oscillations in periods of high volatility. At last, when fitting the model, I left 40 observations out of sample in order to assess the performance of the model when forecasting the close price of BTC/USDT. For this purpose, given that the model forecasts log returns, in order to get everything in terms of the actual price, I have used the formula $P_t = P_0\, exp(\sum_{i=1}^{t} r_t)$ where $r_t$ are the log returns, and plotted the 7 days rolling period one step ahead point forecasts, along with the actual price of the asset and the prediction intervals, calculated using the forecasted conditional standard deviation for each out of sample period, the result can be be appreciated in

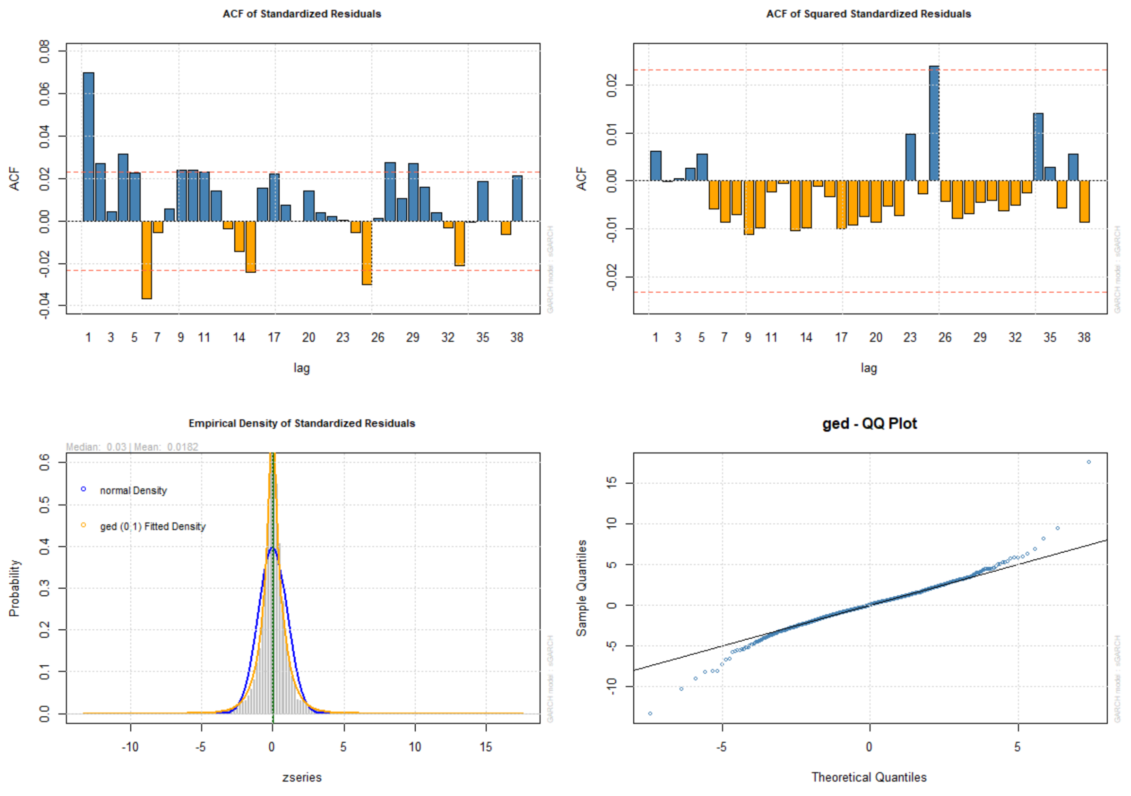


Figure 2.12: Analysis of residuals of ARIMA(2,1,2)-GARCH(1,2)

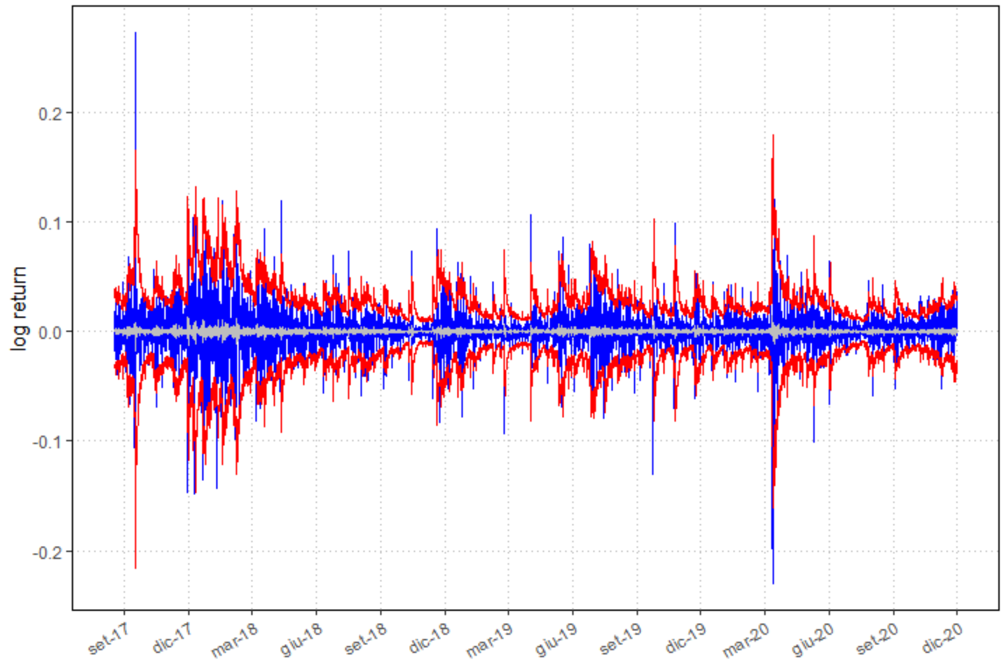


Figure 2.13: Actual returns with fitted standard deviation intervals

figure 2.14. The RMSE and MAE for out of sample price forecasts are, respectively, 184.86 and 146.77. The prediction intervals can be used as a proxy of the next 4 hours volatility of BTC/USDT at each closes and can be useful for a short-term buy-sell strategy where one could buy when the price closes under the lower bound and sell when it reaches the higher bound.

### 2.4.5 Final Takeaways

As seen previously, augmenting a simple ARIMA model with a conditional heteroscedastic GARCH part has its benefits because it provides a useful approximation of the conditional variance that for financial assets tends to persist in some periods, and additionally, the confidence intervals can be used as trading indicators for when to enter and exit the market. However, the model has its own limitations for h-steps ahead forecasts, as both the ARIMA and the GARCH would forecast the constant unconditional mean and variance of the time series due to their assumptions, also, to obtain satisfactory predictions over the long term and to account for unforeseen price behaviours, one has to perform rolling forecasts and refit the model as new observations become available. Moreover, the model would not catch the possible non-linear relations between the most recent observation and the lagged value of the series, so in order to allow for non-linearity, in the next chapter I explored how neural networks can help in this case and to asses their performance when forecasting the price of a financial asset.

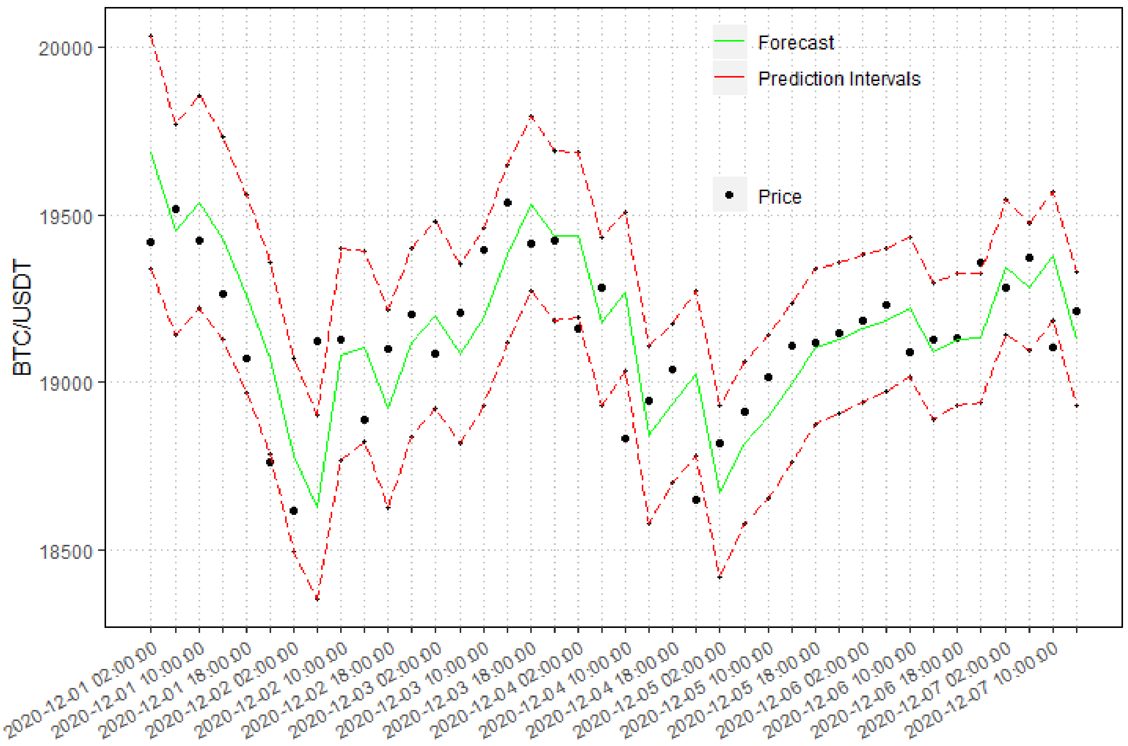


Figure 2.14: Forecasts for the next 7 days out-of-sample

# Chapter 3

# Sequential Learning With Recurrent Neural Networks

Recurrent neural networks are a class of artificial neural networks which belong to the broader discipline of machine learning, the science of programming algorithms to dig into large sets of data with the goal of finding complex and hidden patterns that could be crucial for decision making, problem solving or performing a specific task. Thanks to the technology innovations of the last decade and the ever increasing computational power at our disposal, computers are now capable of handling highly complex algorithms and demanding tasks with such an efficiency and an unprecedented hastiness that could have not been achieved before. The advances in the computer hardware sector, its exponential growth driven by an unparalleled demand, and the availability of huge amounts of data relative to almost every field, have made possible to exploit techniques that never found an actual implementation due to how unpractical and daunting their application would have been, to the extent that, now, new approaches and new disciplines centered around *big data* are now flourishing and improving at a staggering rate. Moreover, the diffusion of computer software and platforms as open source have stirred the interest and commitments of a huge community of developers to contribute in their development and progress, an example being the Silicon valley giant Google that made available to the public, powerful tools, such as Tensorflow[1] and Keras[2], which I have used throughout this chapter, to accomplish a wide variety of machine and deep learning tasks.

### 3.0.1 Supervised Learning

To forecast Bitcoin price, I relied on a supervised sequential learning approach, namely, by partitioning the data set in a training and a testing subsets, the model learns the complex, non-linear

[1]github.com/tensorflow/tensorflow
[2]keras.io/

patterns between the observations by example, and in each training instance predicts the value of the response variable $y_t$ given a sequence of its lagged values, called features or predictors, and recursively adjusts the prediction until it finds the best parameters that minimize the target function which represent the deviation between the output and the actual value of the series, and finally one can assess its forecasting performance in the out of sample test set. The main difference from the other types of machine learning approaches is that, with supervised learning, the variables are already labeled, in this case by the time index, and the human supervision is required to tweak the so called hyperparamters that affect how the learning is performed. Basically, the prediction is based on approximating the non-linear map $F(X_t \mid \theta)$ from an high dimensional matrix of input features to the response variable, where the parameters $\theta$, which are different from the algorithm's hyperparameters, are the degrees of freedom that the model has when approximating the relationship during training. Even if, in this case, the model is parametric, differently from the classical statistical forecasting approach that is based on fitting a theoretical model to the series, neural networks handle model selection and inference in an algorithmic fashion without any assumption on the data generating process, being very flexible when generalizing the structure of a large data set. The main difference from the maximum-likelihood approach seen in the previous chapter, is that now the function that is being minimized is not based on the theoretical distribution of the innovations but on an out-of-sample performance indicator, like the mean squared error or the mean absolute error, specified by the user. Moreover, given the lack of statistical tests to asses the significance of the estimated parameters, in order to optimize the bias-variance trade-off with neural networks, one can rely on many different approaches to avoid overfitting, for example imposing constraints on the parameters with the so called regularization, whose degree is specified by tuning the related hyperparameter at the begging of the training phase.

### 3.0.2 Artificial Neural Networks

The artificial neuron, introduced by Warren McCulloch and Pitts [36], was inspired by the structure of biological neurons and by the hypothesis that an algorithm could emulate their functions and mimic the mechanism through which their components are connected together. In its first instance, the artificial neuron performed simple logical operations with binary output, getting activated if the number of its binary inputs exceeded a specific threshold.

The first artificial neural network, abbreviated ANN, called Perceptron, was invented by Rosenblatt [37], its basic component, the threshold logic unit or TLU, computes a weighted sum of the inputs and applies a *step* function to the result, like the Heaviside function that takes value 1 if its input is positive and zero otherwise. The Perceptron is made of a layer of TLUs, each of them fully connected to the first layer of input neurons such as the network output is $F_{W,b}(X) = \sigma(XW + b)$ where $\sigma$ is the activation function, Heaviside in this case, X is the matrix of inputs NxK and W is the matrix KxH where H is equal to the number of TLUs in the layer, representing the connections of the network, and b is the vector of biases with H columns and takes always value 1 in this case. The perceptron was employed primarily for classification tasks, namely, the network is trained to find the right weights that would make the weighted sum of the inputs positive such that, after applying the activation, the output is 1 if the inputs belong to the target class A, or zero if the inputs do not belong to A.

To overcome the limitations of employing only a single layer of neurons and introduce non-linearity in the decision boundary of the network, a Multilayer Perceptron stacks multiple layers of fully connected neurons such that, between the first pass-trough input layer and the last output layer of neurons there are L hidden layers fully connected to the rest of the network. This architecture, called Deep Neural Network, DNN, is characterized by having L layers of abstraction through which the inputs flow, increasing the computational complexity required to find the complex relationships between the features and the target variables. A deep feedforward neural network, were the inputs X travel sequentially all the layers in the network from the first to the last, can be represented as a composition of functions, namely:

$$\hat{Y}(X) = F_{W,b}(X) = (f^L_{W_L,b_L} \circ \ldots \circ f^1_{W_1,b_1})\,(X) \tag{3.1}$$

where $\hat{Y}(X)$ is the predicted value of the target variable Y, $f^L_{W_L,b_L} = \sigma_L(XW_L + b_L)$ is the output on the $L^{th}$ layer and the weights and biases represents, as before, the connections between each layer of the network, such that there are KxH + H parameters for each layer, and $\sigma_L$ is the activation function that introduces non-linearity between the inputs and the outputs. There is a wide range where activation functions can be chosen from to activate each layer, like for example the hyperbolic tangent, the logistic function or ReLu, each of them fitting particular needs and specific tasks, depending on what the user wants to achieve or which shortcomings of other functions to overcome. It can be easily shown that, if there is only one layer and the activation is the identity function, equation (3.1) is the simple linear regression, demonstrating that a deep feedforward neural network is a non-linear high-dimensional generalization of the classic regression function.

The network learns according to the Backpropagation algorithm, introduced in 1986 by Rumelhart, Hinton, and Williams [38], such that it adjusts the paremeters of each layer iteratively to find the best set that makes the forecasted value closest to the target. Supposing that the generalization error is measured by the mean squared error function, given the set of time-indexed features represented as the NxK matrix X with K columns as the number of features, the Backpropagation algorithm handles the features in batches, 3-dimensional matrices Mx$n$xK where M is the number of sub-matrices, called instances, in the batch and $n \subset N$ are the time steps contained in each instance, the length $M$ can be chosen by the user depending on the amount of RAM available, as the higher the size of the batch the more free space is required in the memory. The algorithm is designed such that, on the first forward pass a batch of instances is sent to the input layer which passes it to the first hidden layer that by computing the weighted sum of the instances and applying the activation function to the result, outputs a matrix, whose entries are in a certain range of

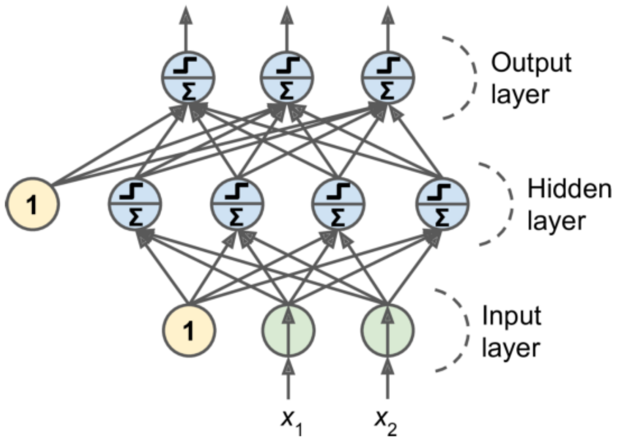


Figure 3.1: Visual representation of a Multilayer Perceptron [3]

values depending on the activation function, which will be the input for the second hidden layer, that repeats the process and sends its output to the next hidden layer and so on, until arriving at the last layer of the network. The latter, called the output layer, in the case of a regression neural network, still calculates the dot product between the matrix of inputs from the preceding layer and the matrix of weights, adding the vector of biases, but does not apply any activation function allowing the results to vary freely without any bounds because the output, which is the predicted value $\hat{Y}$, needs to have the same scale as $Y$ to compute the MSE.

The parameters of the network, the weights and biases, are of fundamental importance because not only they determine the importance given to each feature when predicting $\hat{Y}$, but directly influence the connection between the layers' neurons and whether they are turned on or off, for example, if the activation function is the sigmoid function $\sigma(X) = \frac{1}{1+e^{-X}}$, which is bounded between zero and one, a negative value of the the weighted sum computed by a single neuron in the layer, once passed through the activation function, makes the relative entry of the output matrix equal to zero such that when the latter will be sent as the input for the next layer, its neurons will not connect to it.

In the backward pass, the algorithm starts computing the gradient vector of partial derivatives of the loss function, the MSE in this case, with respect to the parameters of the last layer in order to asses of much each of its parameters contributed to the generalization error and, by the chain rule, does this for every other layers before the last, in retrospection, by propagating backward the gradient vector until arriving at the first hidden layer. To penalize the connections between the neurons that contributed the most to the error, the algorithm tweaks the weights and biases in order to minimize the loss function, by performing the so called Gradient Descent optimization [39] [40].

[3]Source:"Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems", Aurélien Géron (O'Reilly)

Given the MSE function $MSE(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}(X \mid \theta) - Y)^2$ for the whole batch, were $\theta$ is the vector of parameters of the network, the Gradient Descent algorithm searches for the set of parameters that minimize the loss function, namely, the combination of parameter values such that the gradient vector $\bigtriangledown_\theta MSE(\theta)$ is equal to zero. Once the gradient for the first set of parameters is found after the final step of backpropagation, Gradient Descent adjusts the parameters in the direction where their partial derivatives, with respect to the loss function, are negative, in other words, along the path were the loss has the steepest slope, with the goal of reaching its minimum. If the parameters are not optimal, Gradient Descent performs an additional search over the parameter space, triggering another forward-backward pass through the training set, called epoch, and tweaks the parameters according to their gradient:

$$\theta_{next\ step} = \theta - \eta \bigtriangledown_\theta MSE(\theta) \tag{3.2}$$

such that, at every step, if the gradient of a parameter is positive, the algorithm will go in the opposite direction, doing this for a specified number of epochs. The parameter $\eta$ in (3.2) is called the learning rate and its an hyperparameter that dictates the size of the step the gradient descent takes when percurring the loss function, and its fundamental because if set too high, the algorithm will takes too big steps along the loss, bouncing around the minimum and converging to a sub-optimal solution, whereas if set too low, the training process will be more time-consuming. Typically the number of epochs is set to be high enough to let the algorithm find the minimum and the user can employ an early stopping criteria through the *callback* function that stops the iteration when the value of the loss does not improve after n steps, without worrying about wasting time and resources.

To reduce the risk of overfitting the test set, a common approach is to add to the loss function a penalty term accounting for the number of parameters of the model, such that, the ones with too much degrees of freedom would perform worse than more parsimonious specifications at optimizing the bias-variance trade-off. I have chosen to add the $\ell_1$ norm, to not penalize the outliers in the data set too much, to the MSE, such that the loss function takes the form:

$$Loss(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{3.3}$$

where $\alpha$ is an hyperparameter that influences the degree of the penalty.

When employing a very deep neural network with an high number of neurons for each layer, one can incur in the problem of unstable gradients, a symptom that the model is overfitting the data. This is correlated with the activation function chosen for the layers, indeed, taking as an example the sigmoid function again, which has been the most common choice by practitioners, it saturates when the its input is too large or too small and the derivative at its boundary become zero so when there is a large number of neurons in the layers of the network there is the risk that the backpropagation algorithm, which computes the gradients at every step of the training session and propagates them according to the Gradient Descent, would tweak the parameters of the upper layers with an already small value and as it reaches the lower layers, the gradients have already been diluted, so the parameters of the first layers would not receive any change at all. Amongst many approaches and solutions to the unstable gradients, I have chosen to employ the Glorot and Bengio [41] parameters initialization, both because of its efficiency and its compatibility with the activation function of my choice. Indeed, it is essential to initialize the weights and biases in a consistent way and avoid setting their initial values to zero or the same constant value, because that would lead the Backpropagation algorithm to tweak the parameters in the same way for every layer and the result would be like having a neural network with just one neuron.

Glorot and Bengio pointed out that to alleviate the instability of the gradients, the signal in the forward and back passes do not need to be dispersed or grow exponentially large and theoretically, to ensure that, both the inputs and the outputs of the network must have equal variance and at the same way, the gradients flowing in either direction need to have the same variance too, in practice however, this is not feasible, so the authors proposed and efficient compromise, namely, initializing the parameters randomly according to a uniform distribution between -r and r, with $r = \sqrt{\frac{3}{fan_{avg}}}$, where $fan_{avg} = \frac{fan_{in}+fan_{out}}{2}$ is the average between between the number of inputs and neurons of the layers in the network.

Additionally, over the years, the Gradient Descent algorithm has inspired the development of new and faster optimizers, each one of them accommodating various needs depending on the particular use-case or aiming to overcome the limitations of the other optimizers available for training neural networks. The one I chose the network to be trained with is Adam, which stands for adaptive moment estimation, introduced by Kingma and Ba [42], which gathers the functionalities and intuitions of other two optimizers under a single, powerful tool. Specifically, the steps that the algorithm takes for tweaking the parameters are:

$$m = \beta_1 m - (1-\beta_1) \bigtriangledown_\theta Loss(\theta) \tag{3.4}$$

$$s = \beta_2 s - (1-\beta_2) \bigtriangledown_\theta Loss(\theta) \otimes \bigtriangledown_\theta Loss(\theta) \tag{3.5}$$

$$\overline{m} = \frac{m}{1-\beta_1^t} \tag{3.6}$$

$$\overline{s} = \frac{s}{1-\beta_2^t} \tag{3.7}$$

$$\theta_{ns} = \theta + \eta \overline{m} \oslash \sqrt{\overline{s} + \epsilon} \tag{3.8}$$

where $t$ is the number of the iteration and $\eta$ is the learning rate hyperparameter. When updating the weights and biases, Adam takes into account the geometric average of both the past gradients and the square of past gradients. The equation (3.4) defines the momentum [43] of the parameter updates, increasing the acceleration by which the optimizer converges to the minimum of the loss, indeed, thanks to this approach, the hyperparameter $\beta_1$ can be tweaked in such a way that, when the gradients are very small or remain almost constant, while Gradient Descent would start going very slow, the momentum kicks in and let the algorithm escape the plateaus of the loss function faster than the former. Whilst, equation (3.5) lets the learning rate be adaptive [44], indeed, if past gradients are very high in magnitude, scaling down the learning rate by the term $\overline{s}$ makes the optimizer take more short and precise steps when moving along the steepest dimension, fostering the convergence toward the minimum of the loss. To avoid the learning rate decaying too much and too abruptly, the parameter $\beta_2$ controls the speed by which the square of past gradients decays, such that, setting an higher $\beta_2$ makes sure that only the most recent gradients will affect $\eta$ and the parameter updates.

### 3.0.3 Recurrent Neural Networks

Recurrent neural networks [45] [46], RNN, have become increasingly popular in the financial sector, mainly due to their peculiar architecture that makes them well suited for processing long series of data, such that, there has been a rise in adoption for many different purposes, especially for forecasting time series and building speculative trading strategies.

The econometric models presented before do not learn from the data and given their linear specification, often tend to misrepresent the actual relationship between the observations, whereas neural networks, mining through the data iteratively, try to catch as many patterns as possible, and can be considered as an extension of time series models, but with less theoretical assumptions.

Moreover, the strength of RNNs when applied to time series forecasting is that there is recurrence that builds memory in the network. Indeed, as opposed to feed forward neural networks seen before where the input flows in one direction until it reaches the last output layer, in a RNN at each time step, every single neuron after receiving the input, produces its output as before, but now, it also sends the output back to itself, thus a recurrent term is included in the calculation performed by the neurons. In other words, take for example a recurrent neural network with a single recurrent neuron, it is said that the network can be unrolled through time [47], such that, given the matrix $\{X_t\}_{t=1}^{N}$ of features, if the first instance of the batch contains five lagged observations of the features in each column represented as the sequence of vectors $(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-3}, \boldsymbol{x}_{t-4})$, when the recurrent neuron processes the instance, at the first time step it takes $\boldsymbol{x}_{t-4}$ and produces $\hat{\boldsymbol{y}}_{t-4}$, at the second time step, takes as input both $\boldsymbol{x}_{t-3}$ and $\hat{\boldsymbol{y}}_{t-4}$ to output $\hat{\boldsymbol{y}}_{t-3}$ and so on for as many time steps in the instance , until arriving at the last time step t whose output will depend on the outcomes of the preceding time steps due to the recurrent memory of the neuron.[4]

The RNN still learns according to the backpropagation algorithm but now the gradients will flow backward through the unrolled network, this is why for RNNs it is called BPTT [48], backpropagation through time, and the algorithm will add up the parameters of each single time step. Multiple recurrent neurons can be arranged in layers and the network will behave in the same way as described above, feeding the output of each layer back to them, in order to account for recurrence and create memory of the past of the series. A RNN whose last layer outputs a vector of predicted variables for each time step is called a sequence-to-sequence network and the loss

[4]At the start of the recursion, the recurrent term is zero because there is no previous output yet.

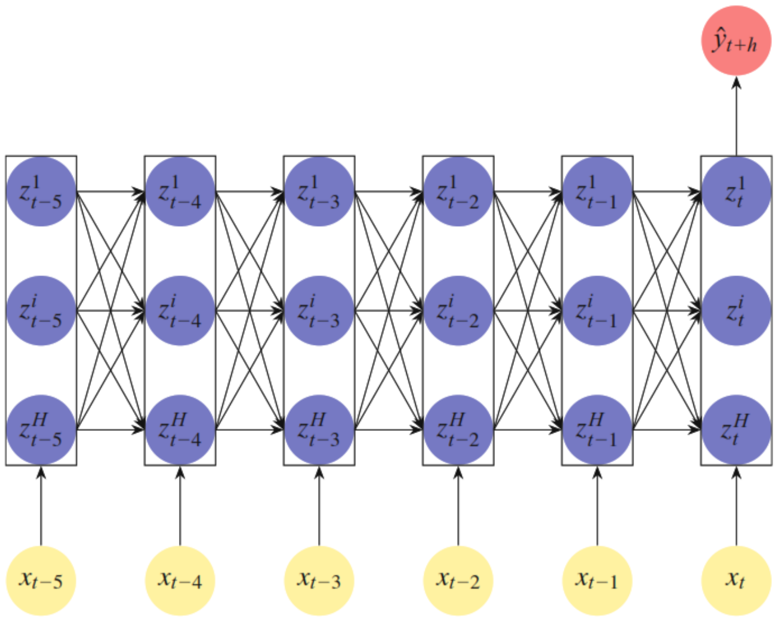


Figure 3.2: Visual representation of a Recurrent Neural Network [5]

function will be calculated taking into consideration the output of every time step, whereas, the network I will employ, called sequence-to-vector, follows a many to one approach, meaning that the layer still unrolls through time and uses the recurrent term at each time step but will effectively output only the vector of predictions of the last time steps such that, the loss function will be computed only for the latter, that is, the vector $\hat{\boldsymbol{y}}_t$.

In the case of univariate time series sequential learning, given the autocorrelated series $\{y_t\}_{t=1}^{N}$ and the vector of lagged observations $\{x_t\}_{t=1}^{N-h}$, where h is the forecast horizon, the approach is to construct a vectorized sequence of lagged values of $y_t$ such that each instance in the batch contains a specified number of lags. Explicitly, each instance is a vector of the form $\{\boldsymbol{x}_{t-j}\}$ where $j = 1, \cdots, T$ are the time steps and T is the look-back period determining the number of lags in the instance. In this context, given that $t = 1, \cdots, N$ and h=1, the batch is built with a sliding window approach, such that, taking as an example T=3, the first instance will contain $\{x_1, x_2, x_3\}$ to forecast $\hat{y}_4$, the next instance is made by $\{x_2, x_3, x_4\}$ and the last time step forecast is $\hat{y}_5$, and so on.

Thus, considering a deep sequence-to-vector RNN with L-1 recurrent layers, the output for a whole batch, in the case of univariate time-series, can be computed as:

$$\hat{y}_t = f^{L}_{W_L,b_L}(Z_t) = \sigma_L(Z_t W_L + b_L) \tag{3.9}$$

$$Z_{t-j} = f^{L-1}_{W_{L-1},b_{L-1}}(\boldsymbol{x}_{t-j}) \tag{3.10}$$

$$= \sigma_{L-1}(Z_{t-j-1}W^{(z)}_{L-1} + \boldsymbol{x}_{t-j}W^{(x)}_{L-1} + b_{L-1}), \qquad j \in \{1, \cdots, T\} \tag{3.11}$$

where (3.9) is the output of the last layer, which in this case is a non-recurrent dense layer with

[5] Source: "Machine Learning in Finance: From Theory to Practice", Dixon, Halperin, Bilokon (Springer)

one neuron such that, $\hat{y}_t$ is the Mx1 vector with M rows equal to the number of instances in the batch, containing the predictions of the last time step for all the instances, and $W_L$ is a vector of size Hx1. $Z_t$ is the hidden state, also called memory cell, of the network, the recurrent term that preserves the memory in the RNN, such that the output of the last time step is a function of all the inputs of the previous time steps, it is an MxH matrix with H equal to the number of neurons in the recurrent layer. $b_L$ is the vector of biases 1xH and $\sigma_L$ is the activation function of the L layer. In (3.11), the term $W^{(z)}$ is the HxH matrix containing the connections between the current input and the output of the last time step, whereas $W^{(x)}$ is the KxH matrix of connections between the input and the layer's neurons, where K=1 in this case, and $\boldsymbol{x}_{t-j}$ is the vector Mx1 of inputs of the current time step. The fist hidden state will depend only on the inputs as, for J=T, the term $Z_{t-T-1}$ is zero. In general, for a layer of this form, there are (K+H)xH+H paramters in total, depending on the number of features, the number of recurrent neurons and biases.

Equation 3.11 shows that the memory of the network depends on the parameter T, the look-back period, and an RNN of this form can be considered as a generalization of an AR(p) process when T is taken to be equal to the autoregressive order p, at each time step the network applies an autoregressive function $f_{W.b}(\boldsymbol{x}_{t-j})$ such that the output will depend on the number of lags in each instance. Indeed, if there layer has only one neuron, the activation function is the identity function, $W^{(z)} = \phi_z$, $W^{(x)} = \phi_x$, $b_{L-1} = 0$, $W_L = 1$ and $b_L = \mu$, an RNN can be represented as an AR(p) of the form $\hat{y}_t = z_{t-1} + \mu$, where $z_{t-1} = \phi_z z_{t-2} + \phi_x x_{t-1}$ and so on, until the first time step where the hidden state depends only on the input $z_{t-p} = \phi_x x_{t-p}$, such that the general form is:

$$\hat{y}_t = \mu + \phi_x(L + \phi_z L^2 + \ldots + \phi_z^{p-1} L^p)x_t \tag{3.12}$$

$$= \mu + \sum_{j=1}^{p} \phi_j x_{t-j} \tag{3.13}$$

with geometrically decaying autoregressive weights $\phi_j = \phi_x \phi_z^{j-1}$ when $|\phi_z| < 1$. Therefore, the number of time steps over which the network unrolls can be determined by looking at the PACF of the series and choosing T to be equal to the last significant lag. Indeed, the PACF of an RNN has the same behaviour of an autoregressive process and cuts off when $h > p$, for example, given an RNN(1) of the form $\hat{y}_t = \sigma(\phi y_{t-1})$, the autocovariance function with h=1 is $\gamma(1) = \mathrm{E}[y_t - \mu, y_{t-1} - \mu]$, given that $y_t = \hat{y}_t + \epsilon_t$, by substitution, $\gamma(1) = \mathrm{E}[\sigma(\phi y_{t-1})\, y_{t-1}]$, with $\mu = 0$, if the activation is the identity then the autocovariance has a form similar to the autocovariance of an AR(1) process, that is, $\gamma(1) = \phi VAR[y_{t-1}]$. When h=2, the autocovariance of an RNN(1) process can be computed as $\gamma(2) = \mathrm{E}[y_t - \hat{y}_t, y_{t-2} - \hat{y}_{t-2}]$, where the hat terms are again the regressions of $y_t$ and $y_{t-2}$ the term in the middle $y_{t-1}$, by approximating $\hat{y}_{t-2}$ as $\hat{y}_{t-2} = \sigma(\phi y_{t-1}) = \sigma(\phi(\hat{y}_{t-1} + \epsilon_{t-1})$, one can see that the latter does not depend on $\epsilon_t$, thus by substitution, the autocovariance at lag 2 becomes $\gamma(2) = \mathrm{E}[\epsilon_t, y_{t-2} - \sigma(\phi(\hat{y}_{t-1} + \epsilon_{t-1})] = 0$, it follows that, like an autoregressive process, the autocovariance, and the PACF, of an RNN(1) cuts off when the lag order is greater then the look-back period and can be useful when specifying the value of the latter.

Additionally, the stationarity condition of an AR(p) process can be generalized in terms of RNNs, such that, with the same assumption as above, the general form of the infinite representation of an RNN(1) is:

$$y_t = \phi(L)^{-1}\,\epsilon_t \tag{3.14}$$

$$= \frac{1}{1-\sigma(W^{(z)}L+b)}\,\epsilon_t \tag{3.15}$$

$$= \sum_{j=0}^{\infty}\sigma^j(W^{(z)}L+b)\,\epsilon_{t-j} \tag{3.16}$$

From (3.16), the stability condition for an RNN(p), similarly the autoregressive counterpart, requires that $|\sigma| < 1$, such that, to ensure that the sum is stable, the common choice is to employ the hyperbolic tangent as the activation function for the layers in the network, defined as $tanh = \frac{e^{2x}-1}{e^{2x}+1}$, which is S-shaped and bounded between -1 and 1. Equation 3.16 shows that the effect of distant lags and of the recurrent terms on the layer output decays geometrically with time.

### 3.0.4 $\alpha$-RNN

The plain RNN has a short memory, which length depends on the parameter T, so in order to allow for a longer memory, the $\alpha$-RNN [49] model adds a smoothing parameter to the hidden states, such that:

$$\hat{y}_{t+1} = z_t W + b \tag{3.17}$$

$$z_t = \sigma(\tilde{z}_{t-1}W_L^{(z)} + x_t W_L^{(x)} + b_L) \tag{3.18}$$

where $\hat{y}_{t+1}$ is the predicted value of a single instance, W is the vector Hx1 of connections of the output layer, $z_t$ is the vector 1xH of hidden states, the size of $W_L^{(z)}\,W_L^{(x)}$ are respectively HxH 1xH and $\tilde{z}_{t-1} = \alpha z_{t-1} + (1-\alpha)\tilde{z}_{t-2}$, where at the start of each instance, $z_{t-T+1} = \phi x_{t-T+1}$. Differently from an RNN, the $\alpha$ in this case provides an infinite memory to the model. Indeed, by considering all the biases equal to zero, W=0, $W^{(z)} = W^{(x)} = \phi$, $\sigma = Id$ and T=2, (3.18) becomes $z_t = \phi(\alpha\phi x_{t-1} + (1-\alpha)\tilde{z}_{t-2} + x_t)$ and (3.17) can be re-written as:

$$\hat{y}_{t+1} = \phi x_t + \alpha\phi^2 x_{t-1} + \phi(1-\alpha)\tilde{z}_{t-2} \tag{3.19}$$

such that when $\alpha = 1$ the model is a plain RNN, otherwise, the model has infinite memory, and $\tilde{z}_{t-2}$ depends on the starting point of the series $x_1$.

### 3.0.5 Data Transformation & Model Identification

First, I split the data set into a training and test set according to a ratio of 90/10. Afterwards, given that the sample covers roughly two years of trading, the close prices of two distant periods of time can be very different in magnitude, so in order to avoid the risk of unstable gradients, I scaled all the observations in both the training and test set by subtracting the mean and dividing by the standard deviation computed over the training set, in order to avoid a look-ahead bias.

In order to fix the look-back parameter T, I have checked the PACF of the the training set, and as can be seen from the figure below, there is a spike at lag 6 out of the confidence bounds, thus I set T=6, namely, the batch will contain vectorized sequences of 6 lags each and is constructed with a sliding window approach, as discussed before, with a forecast horizon h=1.

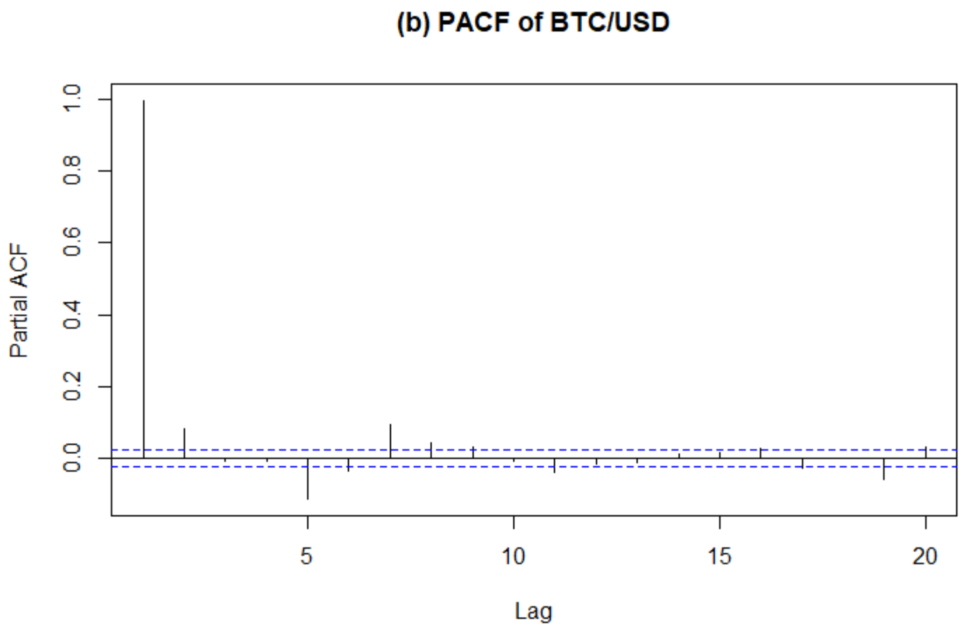


Figure 3.3: PACF of the training set

By using *Scikit-learn*, *Keras* and *Tensorflow* libraries in Python, I choose to employ an $\alpha$-RNN with one hidden layer, to avoid both exploding/vanishing gradients and overfit the sample, a dense, single neuron, layer as the output layer, activation function *tanh*, Glorot and Bengio parameters initializers, Adam optimizer and specified the mean squared error as the loss function for the training instance.

In order to find how many neurons to include in the layer and which parameter for the $\ell_1$ penalty to use, I performed a grid search over the hyperparameter space with *GridSearchCV*, which evaluates the performance of the model on the test set for every combination of the target hyperparameters. The results are then cross-validated using a 5-fold cross-validation, namely, by splitting the training set into 5 pairs of training and test subsets with the *TimeSeriesSplit* function, where the test set is always ahead in time, the algorithm returned the pair of parameters that performed the best on average, which was H=110 for the number of neurons and $\alpha_{\ell_1} = 0$.

### 3.0.6 Training and Results

Using the pair just found, I let the model train for 2000 epochs and fixed an *early stopping* criteria. From the summary of the model, the number of estimated parameters in the first layer are equal to (1+110)x110+110+1, taking into account also the $\alpha$ hyperparameter, estimated to be equal to 0.17.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
alpha_rnn_7 (AlphaRNN)      (None, 110)               12321
_________________________________________________________________
dense_7 (Dense)             (None, 1)                 111
=================================================================
Total params: 12,432
Trainable params: 12,432
Non-trainable params: 0
_________________________________________________________________
```

The mean squared error is lower in the training set than in the test set, as expected, and after transforming the data back to its original scale, I have plotted the fitted values of the model against the training set to grasp its performance during training.

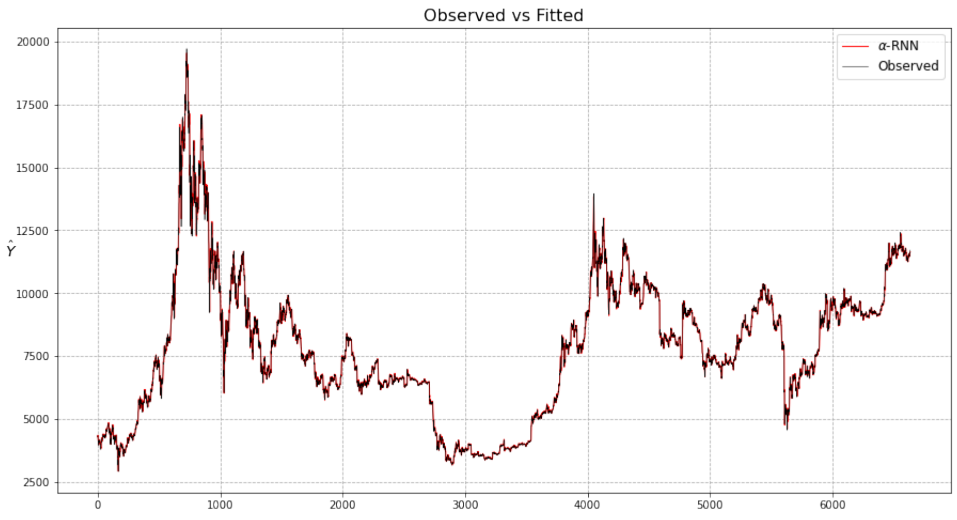


Figure 3.4: Performance of the model in the training set

However, the performance in the out-of-sample set, which is the same used in the previous chapter, is completely different, as showed in figure 3.5, and exposes the limitations of the model in approximating the highly volatile behaviour of the series. The rolling forecasts of $y_t$ based on its 6 previous values, are clearly lagging by one period, reducing the actual suitability of constructing a speculative trading strategy around the model, the out-of-sample RMSE is 188.11 and the MAE is equal to 137.38.

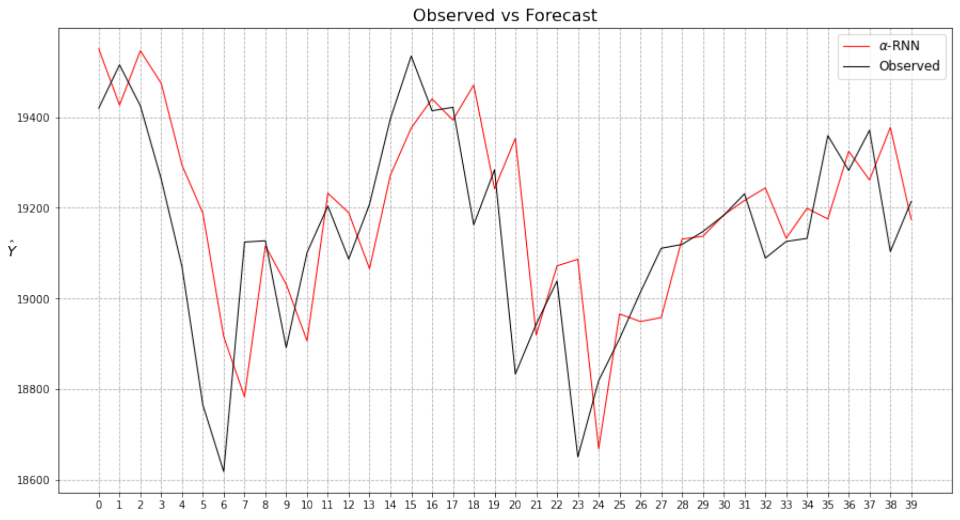


Figure 3.5: Out-of-sample rolling, one period ahead forecasts

These findings are coherent with the characteristics of the model employed, in fact, the weights $W_L^{(z)}$ and $W_L^{(x)}$ in an RNN are time-invariant and do not change between the time-steps when a layer unfolds, so the performance of the model degrades when used to approximate the non-linear behaviour of an autocorrolated non-stationary time series like the one in this case. Moreover, given that an RNN generalizes an AR(p) model, it expects the innovations to be i.i.d. white noise, thus it performs better in case of homoscedastic errors. In order to overcome this shortcoming, an alternative approach would be to extend the simple RNN with a Generalized Recurrent Neural Network, GRNN, that similarly to the GARCH extension of an autoregressive model, is better suited to model the hetetoscedasticity that often financial time series exhibit.

The low value of the fitted $\alpha$ parameter shows that, in order to optimize the loss function, the model needs to account for long-term relationships and distant past lagged values of the target, so in order to obtain a better forecasting performance, one could employ a recurrent neural network with a differently structured memory cell, like LSTMs and GRUs, to enhance both the memory in the network and the method by which it handles past observations, and obtain a better approximation of the price dynamic while also accounting for more complex patterns occurred in past trading days.

# Conclusions

In the first chapter of this thesis I have outlined the main technical characteristics of the Bitcoin technology and the innovative features that stirred the interest of a large community of tech enthusiasts, programmers and cypherpunks, while driving many retail and institutional investors to increase their stake in the cryptocurrency. A comparative analysis of returns with respect to other traditional financial assets revealed the highly speculative and volatile nature of the market for Bitcoin, so in the second chapter, I have fitted an econometric model to its time series to asses if the forecasts obtained thereby can be effectively used for a speculative trading strategy. At first, the log price of BTC/USDT seemed like an integrated process of order one, so I assumed that a first order difference was needed to meet the stationarity assumption of the theoretical models. Moreover, the residuals of a plain ARIMA model fit, and the ACF of squared residuals, revealed the presence of a persistent conditional heteroscedasticity, so I decided to extend the model with GARCH, and by maximum-likelihood estimation, I found that an ARIMA(2,1,2)-GARCH(1,2), along with the assumption of GED distributed standardized innovations, provided the best compromise in terms of bias-variance compared to the other candidate specifications.

The rolling out-of-sample forecasts revealed that the prediction intervals computed with the standard deviation forecasts of the GARCH part could actually provide an adequate approximation of the volatility of the series and opened to the possibility of back-testing the performance of a buy-sell strategy to asses the long-run viability of using the intervals as a trading indicator.

In the final chapter I have explored recurrent neural networks in order to see if an $\alpha$-RNN model could be able to recognize the complex patterns in the training set and predict an autocorrelated time series by fitting the non-linear relationships between the target, the current value of the series, and the features, six lagged observations. The out-of-sample forecasts were mostly lagging by one period and the model did not perform much better than the econometric linear counterpart, signaling that the theoretical assumptions of the RNN are too strict for a non-stationary series. The results purport that there are ample margins of improvement and further testing is needed to asses if different standardization approaches, adding more layers and neurons to the model, employing measures to reduce the possibility of overfitting the test set, or using other kinds of neural networks, can provide more consistent predictions.

In conclusion, the main difference that emerged from my study is that, while an ARIMA-

GARCH model can be limited by its linear structure and by the paradigms of the classic statistical inference and the related theoretical assumptions, it is precisely from the latter that one can obtain a more interpretable and parsimonious model, whereas, even if recurrent neural networks can take advantage of non-linear functions and the flexibility of cutting-edge architectures, often, to obtain a satisfactory result one has to add more layers of abstraction and increase the parameters to be estimated, thus the overall complexity of the model. In this case, training the model would become increasingly demanding both in terms of time and computer power and one would need to have access to powerful machines in order to deploy a profitable trading strategy in the long run, given that retraining the model frequently is almost necessary to account for new price patterns in the market.

# References

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* (31 October 2008). URL: https://bitcoin.org/bitcoin.pdf.

[2] Antony Lewis. *The Basics of Bitcoins and Blockchains: An Introduction to Cryptocurrencies and the Technology that Powers Them.* Mango, (20 Sep 2018). ISBN: 978-1633538009.

[3] Adam Back. *Hashcash - A Denial of Service Counter-Measure.* (1st August 2002). URL: http://www.hashcash.org/papers/hashcash.pdf.

[4] Antony Lewis. *A Gentle Introduction to Blockchain Technology.* (Sep 9, 2015). URL: https://bitsonblocks.net/2015/09/09/gentle-introduction-blockchain-technology/.

[5] Satoshi Nakamoto. *Bitcoin open source implementation of P2P currency.* (2009-02-11). URL: http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source.

[6] Chris Baraniuk. *Bitcoin's global energy use 'equals Switzerland.* (3 July 2019). URL: https://www.bbc.com/news/technology-48853230.

[7] Paul Roberts. *This Is What Happens When Bitcoin Miners Take Over Your Town - Eastern Washington had cheap power and tons of space. Then the suitcases of cash started arriving.* (9 March 2018). URL: https://www.politico.com/magazine/story/2018/03/09/bitcoin-mining-energy-prices-smalltown-feature-217230.

[8] Robert Sams. *Rehypothecation, Deflation, Inelastic Money Supply and Altcoins.* (August 20, 2014). URL: http://www.ofnumbers.com/2014/08/20/robert-sams-on-rehypothecationde%EF%AC%82ation-inelastic-money-supply-and-altcoins/.

[9] E. Ferraz. *Send Home Your Wages Using Bitcoin and Avoid Hefty Money Transfer Fees? That's Now a Reality.* (3 July 2014). URL: www.techinasia.com/send-home-wages-bitcoin-avoidhefty-money-transfer-fees-reality.

[10] L. Buenaventura. *The Rise of Rebittance: Reinventing Money Transfers in the Philippines with Bitcoin.* (28 September 2014). URL: http://thenextweb.com/insider/2014%20/09/28/rise-rebittance-reinventing-money-transfers-philippinesbitcoin.

[11] F. R. Velde. *Bitcoin: A Primer.* (December 2013). URL: www.chicagofed.org/digital_assets/publications%20/chicago_fed_letter/2013/cfldecember2013_317.pdf.

[12] Lawrence H. White. *The Market for Cryptocurrencies.* (2015). URL: `https://www.cato.org/sites/cato.org/files/serials/files/cato-journal/2015/5/cj-v35n2-13.pdf`.

[13] George E. P. Box et al. *Time Series Analysis: Forecasting and Control.* John Wiley  Sons, (7 August 2015). ISBN: 978-1118675021.

[14] Ruey S. Tsay. *Analysis of Financial Time Series.* John Wiley  Sons, (4 August 2010). ISBN: 978-0470414354.

[15] H. Wold. *A Study in the Analysis of Stationary Time Series.* Almqvist and Wiksell Book Co., Uppsala, (1954).

[16] James D. Hamilton. *Time Series Analysis.* Princeton Univ Press, (30 March 1994). ISBN: 978-0691042893.

[17] David Ruppert and David S. Matteson. *Statistics and Data Analysis for Financial Engineering: with R Examples.* Springer Nature, (31 May 2015). ISBN: 978-1493926138.

[18] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples.* Springer Nature, (22 March 2017). ISBN: 978-3319524511.

[19] Robert F. Engle. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In: *Econometrica* Vol.50.No.4 (July 1982), pp. 987–1007. DOI: `https://doi.org/10.2307/1912773`.

[20] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* Vol.31.Issue 3 (April 1986), pp. 307–327. DOI: `https://doi.org/10.1016/0304-4076(86)90063-1`.

[21] Christian Francq and Jean-Michel Zakoian. *GARCH Models: Structure, Statistical Inference and Financial Applications.* Wiley, (March 2019). ISBN: 978-1-119-31348-9.

[22] Gabriele Fiorentini, Giorgio Calzolari, and Lorenzo Panattoni. "Analytic Derivatives and the Computation of Garch Estimates". In: *Journal of Applied Econometrics* Vol.11.No.4 (Jul. - Aug., 1996), pp. 399–417. DOI: `https://www.jstor.org/stable/2284932`.

[23] Cramér and Harald. *Mathematical Methods of Statistics.* Princeton Univ. Press, (1946). ISBN: 0-691-08004-6.

[24] Rao and Calyampudi Radakrishna. "Information and the accuracy attainable in the estimation of statistical parameters". In: *Bulletin of the Calcutta Mathematical Society* Vol.37 (1945), pp. 81–89. DOI: `https://mathscinet.ams.org/mathscinet-getitem?mr=0015748`.

[25] Carlos M. Jarque and Anil K. Bera. "A Test for Normality of Observations and Regression Residuals". In: *International Statistical Review / Revue Internationale de Statistique* Vol.55.No.2 (Aug., 1987), pp. 163–172. DOI: `https://doi.org/10.2307/1403192`.

[26] S. S. Shapiro and M. B. Wilk. "An Analysis of Variance Test for Normality (Complete Samples)". In: *Biometrika* Vol.52.No.3/4 (Dec., 1965), pp. 591–611. DOI: https://doi.org/10.2307/2333709.

[27] John H. Cochrane. *Asset Pricing*. Princeton Univ. Press, (18 Feb. 2005). ISBN: 978-0691121376.

[28] David A. Dickey and Wayne A. Fuller. "Distribution of the Estimators for Autoregressive Time Series With a Unit Root". In: *Journal of the American Statistical Association* Vol.74.No.366 (Jun., 1979), pp. 427–431. DOI: https://doi.org/10.2307/2286348.

[29] P. C. B. Phillips and P. Perron. "Testing for a Unit Root in Time Series Regression". In: *Biometrika* Vol.75.No.2 (1988), pp. 335–346. DOI: https://doi.org/10.1093%2Fbiomet%2F75.2.335.

[30] D. Kwiatkowski et al. "Testing the null hypothesis of stationarity against the alternative of a unit root". In: *Journal of Econometrics* Vol.54.No.1-3 (1992), pp. 159–178. DOI: https://doi.org/10.1016%2F0304-4076%2892%2990104-Y.

[31] H Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* Vol.19.No.6 (1974), pp. 716–723. DOI: https://doi.org/10.1109%2FTAC.1974.1100705.

[32] Gideon E. Schwarz. "Estimating the dimension of a model". In: *Annals of Statistics* Vol.6.No.2 (1978), pp. 461–464. DOI: https://doi.org/10.1214%2Faos%2F1176344136.

[33] G. M. Ljung and G. E. P. Box. "On a Measure of a Lack of Fit in Time Series Models". In: *Biometrika* Vol.65.No.2 (1978), pp. 297–303. DOI: https://doi.org/10.1093%2Fbiomet%2F65.2.297.

[34] J. Durbin and G. S. Watson. "Testing for Serial Correlation in Least Squares Regression, II". In: *Biometrika* Vol.38.No.1-2 (1951), pp. 159–179. DOI: https://doi.org/10.1093%2Fbiomet%2F38.1-2.159.

[35] Karl Pearson. "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *Philosophical Magazine* Vol.50.No.302 (1900), pp. 157–175. DOI: https://doi.org/10.1093%2Fbiomet%2F65.2.297.

[36] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biology* Vol.5.No.4 (1943), pp. 115–113. DOI: https://doi.org/10.1007/BF02478259.

[37] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological review* Vol.65.No.6 (1958), pp. 386–408. DOI: https://psycnet.apa.org/doi/10.1037/h0042519.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* Ed. by David E. Rumelhart and James L. Mcclelland. MIT Press, 1986. Chap. 8, pp. 318–362.

[39] Claude Lemaréchal. “Cauchy and the Gradient Method”. In: *Documenta Mathematica* Extra Volume: Optimization Stories (2012), pp. 251–254.

[40] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[41] Xavier Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics* Vol.9 (2010), pp. 249–256. URL: `http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf`.

[42] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: `http://arxiv.org/abs/1412.6980`.

[43] Boris Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *Ussr Computational Mathematics and Mathematical Physics* 4 (Dec. 1964), pp. 1–17. DOI: `10.1016/0041-5553(64)90137-5`.

[44] Geoffrey Hinton and Tijmen Tieleman. “Coursera class on neural networks”. In: lecture 6 (2012), slide 29. URL: `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

[45] Jeffrey L. Elman. “Finding Structure in Time”. In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: `https://doi.org/10.1207/s15516709cog1402\_1`.

[46] Michael I. Jordan. “Chapter 25 - Serial Order: A Parallel Distributed Processing Approach”. In: *Neural-Network Models of Cognition.* Ed. by John W. Donahoe and Vivian Packard Dorsel. Vol. 121. Advances in Psychology. North-Holland, 1997, pp. 471–495. DOI: `https://doi.org/10.1016/S0166-4115(97)80111-2`.

[47] Aurélien Géron. *Hands-On Machine Learning With Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems.* Oreilly Associates Inc, (11 October 2019). ISBN: 978-1492032649.

[48] P. J. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: `10.1109/5.58337`.

[49] Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance: From Theory to Practice.* Springer, (1 July 2020).

# Summary

In late 2008, an unknown person, or perhaps a group of people, under the pseudonym of Satoshi Nakamoto posted on the website *bitcoin.org* a white paper called "Bitcoin: a Peer-to-Peer electronic Cash System" [1], proposing a decentralized network that would allow its participants to send and receive digital currency directly, without the need of an intermediary. The role of the trusted third party is replaced by a cryptographic algorithm that builds confidence between the participants of the network, acting as a proof that the transactions are genuine and the funds transferred exists, avoiding Bitcoins counterfeiting, a problem called *double-spending*. All the transactions between the peers are immutably recorded in blocks, and every new transaction is uniquely linked to the previous ones contained in earlier blocks by an 'hash' ID, thus forming a chain of blocks. This chain, called the blockchain, is the ledger where the information about the entire history of all the transactions that ever happened in the network is recorded on. The peculiarity of this electronic ledger lies in its distributed nature as anyone can become a 'node' of the network by downloading the open-source bitcoin client and start participating in the network.

Bitcoin circulating supply is capped at 21 million and geometrically decays with time as designed by its protocol. Its innovative characteristics propagate confidence and trust between its users, bolstering its effective utility. Additionally, the pre-programmed supply limit has deflationary effects, thus helping to maintain the value of Bitcoin in the long-run due to its inevitable scarcity as long as demand for it stay constant or increases. However, citing Robert Sams, "the downside of a known, predictable, and completely inelastic supply unrelated to a fluctuating demand results in perpetual price volatility" [8].

Bitcoin can be traded on exchanges with the ticker BTC and additionally to BTC-fiat currency pairs, it can be traded against *stablecoins*, digital tokens that are programmed to be pegged to a specific fiat currency, like TetherUSD (USDT).

I have applied ARIMA and GARCH econometric models to perform a univariate time series analysis and prediction of Bitcoin price series. The data set in analysis comprises the historical 4 hour close prices of BTC/USDT from 2017-08-17 to 2020-11-30, downloaded with Python, through the API of Binance exchange.

In order to stabilize the variance and induce some symmetry in the data I have applied a log transformation to the time series. The log-prices exhibit a behaviour similar to a random-walk, so

it might be considered as an $I(1)$ and, by taking a first order difference, the process shows mean-reversion around zero. To check if log returns are stationary, I run the augmented Dickey-Fuller [28], Phillips-Perron [29] and KPSS [30] tests. The output support the hypothesis of stationarity with satisfactory p-values. To asses the goodness of fit while still accounting for model complexity, I relied on the Akaike's [31] and Bayesian information criteria [32]

The best candidate models were ARIMA(2,1,2) and ARIMA(2,1,3). The residuals exhibit a consistent volatility clustering and heavy tails when compared to a normal distribution, so employing a GARCH model might explain this behaviour more appropriately. The ACF of squared residuals is decaying rather slowly, persisting across all lags, and hardly respects the confidence intervals, signaling the presence of conditional heteroscedasticity.

By allowing the constant innovations to follow an autoregressive process instead, I tested if a GARCH(p,q) model would fit the conditional standard deviation of the series. Employing the *rugarch* package in R and considering both models for the conditional mean found before, the optimal specification was ARIMA(2,1,2)-GARCH(1,2) with GED distributed standardized residuals. The latter was the model with the most statistically significant parameters and offered the best bias-variance trade-off. The ACF of the squared residuals is inside the confidence bounds for all short-term lags and both the Q-Q and the density plots show that the assumption of GED on stadardized residuals provides a better fit than the Gaussian distribution.

I plotted the actual log-returns of BTC/USDT with the fitted conditional standard deviation superimposed and the fitted conditional mean of the time series. The GARCH part performs well at fitting the conditional heteroscedasticity and provides a satisfying approximation of the wide scope of oscillations in periods of high volatility.

Figure 3.6 reports the out of sample rolling, one step ahead, point forecasts, along with the actual price of the asset and the prediction intervals, calculated using the forecasted conditional standard deviation. The RMSE and MAE for out of sample price forecasts are, respectively, 184.86 and 146.77. The prediction intervals can be used as a proxy of the next 4 hours volatility of BTC/USDT at each closes and can be useful for a short-term buy-sell strategy where one could buy when the price closes under the lower bound and sell when it reaches the higher bound.

However, the model has some limitations for h-steps ahead forecasts, as both the ARIMA and the GARCH would forecast the constant unconditional mean and variance of the time series due to their assumptions, also, to obtain satisfactory predictions over the long term and to account for unforeseen price behaviours, one has to perform rolling forecasts and refit the model as new observations become available. Moreover, the model would not catch the possible non-linear relations between the most recent observation and the lagged values of the series, so in order to allow for non-linearity, I explored if neural networks can help in forecasting the price of a financial asset.

To forecast Bitcoin price, I relied on a supervised sequential learning approach, namely, by partitioning the data set into training and testing subsets, the model learns the non-linear patterns

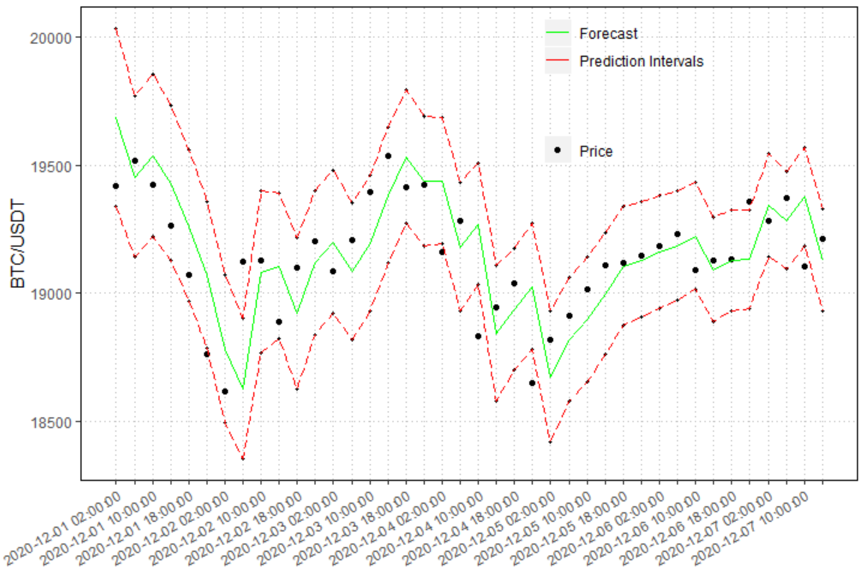


Figure 3.6: Forecasts for the next 7 days out-of-sample

between the observations by example. In each training instance the model predicts the value of the response variable $y_t$ given a sequence of its lagged values, called features or predictors, and recursively adjusts its parameters in order to minimize the loss function representing the deviation between the output and the actual value of the series, and finally assesses the forecasting performance in the out of sample test set.

The network learns according to the Backpropagation algorithm [38]. The algorithm handles the features in batches, 3-dimensional matrices Mx$n$xK where M is the number of sub-sequences in the batch, called instances, $n$ are the time steps contained in each instance and K is the number of features. On the first forward pass a batch is sent through the layers of the network until arriving at the last output layer where the algorithm computes loss function. In the backward pass, the algorithm starts computing the gradient vector of partial derivatives of the loss function with respect to the parameters, the weights and biases of the last layer and, by the chain rule, does this for every other layers before the last, in retrospection, by propagating backward the gradient vector until arriving at the first hidden layer. The algorithm tweaks the weights and biases by performing a Gradient Descent optimization [39] [40]. The latter adjusts the parameters along the path were the loss has the steepest slope.

In Recurrent neural networks [45] [46], differently from feed forward neural networks, at each time step, every layer sends its output back to itself, it is said that the network can be unrolled through time [47]. Thus a recurrent term, called hidden state, is included in the calculations performed by the neurons to create memory in the RNN, such that the output of the last time step is a function of all the inputs of the previous time steps. The RNN still learns according to the backpropagation algorithm but now the gradients will flow backward through the unrolled network, this is why for RNNs it is called BPTT [48], backpropagation through time, and the algorithm will add up the parameters of each single time step.

In the case of univariate time series sequential learning, the batch is built with a sliding window approach, such that, each instance in the batch contains lagged values of the target variable. The memory length of the network depends on the look-back parameter which specifies the number of lags in the instances.

A simple RNN can be considered as a generalization of an AR(p) process when the look-back is equal to the autoregressive order p. Therefore, the PACF of an RNN has the same behaviour of an autoregressive process and can be useful when specifying the value of the look-back parameter. Additionally, the stationarity condition of an AR(p) process can be generalized in terms of RNNs, such that, the stability condition for an RNN(p) is $|\sigma| < 1$, where $\sigma$ is the activation function of the network's layers and induces non-linearity. To respect this condition, the common choice is to employ the hyperbolic tangent as the activation function.

In order to allow for a longer memory, the $\alpha$-RNN [49] model adds a smoothing parameter to the hidden states. When $\alpha = 1$ the model is a plain RNN, otherwise, the model has infinite memory, and the hidden states depend on the starting point of the whole series.

After splitting the data set, I scaled all the observations in both the training and test sets by subtracting the mean and dividing by the standard deviation computed over the training set, in order to avoid a look-ahead bias. The PACF of the training set exhibited a significant spike at lag six, hence I set the look-back to be equal to 6 such that the batch will contain vectorized sequences of 6 lags each. By using *Scikit-learn*, *Keras* and *Tensorflow* libraries in Python, I choose to employ an $\alpha$-RNN with one hidden layer, to avoid both exploding/vanishing gradients and overfit the sample, a dense, single neuron, layer as the output layer, activation function *tanh*, Glorot and Bengio parameters initializers [41], Adam optimizer, which is a modified version of Gradient Descent, and specified the mean squared error as the loss function for the training instance. In order to find how many neurons to include in the layer, I performed a grid search with *GridSearchCV*. The number of neurons that performed the best on average was 110. Afterward, I let the model train, the total number of estimated parameters was 2,432 and $\alpha$ was estimated to be equal to 0.17.

The Out-of-sample rolling, one step ahead, forecasts (Fig. 3.7) of $y_t$ based on its 6 previous values are mostly lagging by one period, signaling that there are ample margins of improvement and further testing is needed to asses if different standardization approaches, adding more layers and neurons to the model, employing measures to reduce the possibility of overfitting the test set, or using other kinds of neural networks, can provide more consistent predictions. The out-of-sample RMSE is 188.11 and the MAE is equal to 137.38.

The results are coherent with the characteristics of the model employed, in fact, the weights in an RNN are time-invariant and do not change between the time-steps when a layer unfolds, so the performance of the model degrades when used for a non-stationary time series. Moreover,

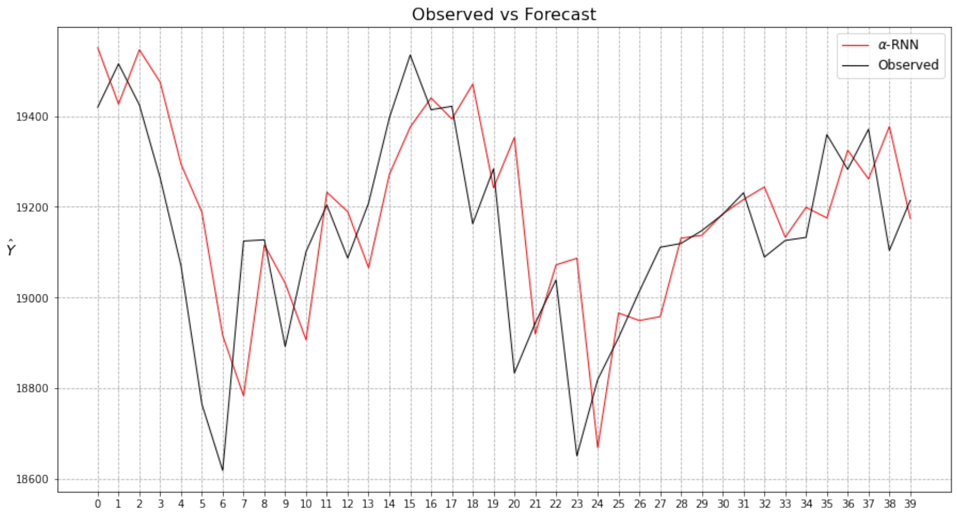


Figure 3.7: Out-of-sample rolling, one period ahead forecasts

given that an RNN generalizes an AR(p) model, it expects the innovations to be i.i.d. white noise, thus it performs better in case of homoscedastic errors. In order to overcome this shortcoming, an alternative approach would be to extend the simple RNN with a Generalized Recurrent Neural Network, GRNN, that similarly to the GARCH extension of an autoregressive model, is better suited to model hetetoscedasticity.

The low value of the fitted $\alpha$ parameter shows that, in order to optimize the loss function, the model needs to account for long-term relationships and distant past lagged values of the target, so in order to obtain a better forecasting performance, one could employ a recurrent neural network with a differently structured memory cell, like LSTMs and GRUs, to enhance both the memory in the network and the method by which it handles past observations, and obtain a better approximation of the price dynamic while also accounting for more complex patterns occurred in past trading days.

In conclusion, the main difference that emerged from my study is that, while an ARIMA-GARCH model can be limited by its linear structure and by the paradigms of the classic statistical inference and the related theoretical assumptions, it is precisely from the latter that one can obtain a more interpretable and parsimonious model, whereas, even if recurrent neural networks can take advantage of non-linear functions and the flexibility of cutting-edge architectures, often, to obtain a satisfactory result one has to add more layers of abstraction and increase the parameters to be estimated, thus the overall complexity of the model. In this case, training the model would become increasingly demanding both in terms of time and computer power and one would need to have access to powerful machines in order to deploy a profitable trading strategy in the long run, given that retraining the model frequently is almost necessary to account for new price patterns in the market.

# Bibliography

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* (31 October 2008). URL: `https://bitcoin.org/bitcoin.pdf`.

[2] Antony Lewis. *The Basics of Bitcoins and Blockchains: An Introduction to Cryptocurrencies and the Technology that Powers Them.* Mango, (20 Sep 2018). ISBN: 978-1633538009.

[3] Adam Back. *Hashcash - A Denial of Service Counter-Measure.* (1st August 2002). URL: `http://www.hashcash.org/papers/hashcash.pdf`.

[4] Antony Lewis. *A Gentle Introduction to Blockchain Technology.* (Sep 9, 2015). URL: `https://bitsonblocks.net/2015/09/09/gentle-introduction-blockchain-technology/`.

[5] Satoshi Nakamoto. *Bitcoin open source implementation of P2P currency.* (2009-02-11). URL: `http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source`.

[6] Chris Baraniuk. *Bitcoin's global energy use 'equals Switzerland.* (3 July 2019). URL: `https://www.bbc.com/news/technology-48853230`.

[7] Paul Roberts. *This Is What Happens When Bitcoin Miners Take Over Your Town - Eastern Washington had cheap power and tons of space. Then the suitcases of cash started arriving.* (9 March 2018). URL: `https://www.politico.com/magazine/story/2018/03/09/bitcoin-mining-energy-prices-smalltown-feature-217230`.

[8] Robert Sams. *Rehypothecation, Deflation, Inelastic Money Supply and Altcoins.* (August 20, 2014). URL: `http://www.ofnumbers.com/2014/08/20/robert-sams-on-rehypothecationde%EF%AC%82ation-inelastic-money-supply-and-altcoins/`.

[9] E. Ferraz. *Send Home Your Wages Using Bitcoin and Avoid Hefty Money Transfer Fees? That's Now a Reality.* (3 July 2014). URL: `www.techinasia.com/send-home-wages-bitcoin-avoidhefty-money-transfer-fees-reality`.

[10] L. Buenaventura. *The Rise of Rebittance: Reinventing Money Transfers in the Philippines with Bitcoin.* (28 September 2014). URL: `http://thenextweb.com/insider/2014%20/09/28/rise-rebittance-reinventing-money-transfers-philippinesbitcoin`.

[11] F. R. Velde. *Bitcoin: A Primer.* (December 2013). URL: `www.chicagofed.org/digital_assets/publications%20/chicago_fed_letter/2013/cfldecember2013_317.pdf`.

[12] Lawrence H. White. *The Market for Cryptocurrencies.* (2015). URL: `https://www.cato.org/sites/cato.org/files/serials/files/cato-journal/2015/5/cj-v35n2-13.pdf`.

[13] George E. P. Box et al. *Time Series Analysis: Forecasting and Control.* John Wiley Sons, (7 August 2015). ISBN: 978-1118675021.

[14] Ruey S. Tsay. *Analysis of Financial Time Series.* John Wiley Sons, (4 August 2010). ISBN: 978-0470414354.

[15] H. Wold. *A Study in the Analysis of Stationary Time Series.* Almqvist and Wiksell Book Co., Uppsala, (1954).

[16] James D. Hamilton. *Time Series Analysis.* Princeton Univ Press, (30 March 1994). ISBN: 978-0691042893.

[17] David Ruppert and David S. Matteson. *Statistics and Data Analysis for Financial Engineering: with R Examples.* Springer Nature, (31 May 2015). ISBN: 978-1493926138.

[18] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples.* Springer Nature, (22 March 2017). ISBN: 978-3319524511.

[19] Robert F. Engle. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In: *Econometrica* Vol.50.No.4 (July 1982), pp. 987–1007. DOI: `https://doi.org/10.2307/1912773`.

[20] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* Vol.31.Issue 3 (April 1986), pp. 307–327. DOI: `https://doi.org/10.1016/0304-4076(86)90063-1`.

[21] Christian Francq and Jean-Michel Zakoian. *GARCH Models: Structure, Statistical Inference and Financial Applications.* Wiley, (March 2019). ISBN: 978-1-119-31348-9.

[22] Gabriele Fiorentini, Giorgio Calzolari, and Lorenzo Panattoni. "Analytic Derivatives and the Computation of Garch Estimates". In: *Journal of Applied Econometrics* Vol.11.No.4 (Jul. - Aug., 1996), pp. 399–417. DOI: `https://www.jstor.org/stable/2284932`.

[23] Cramér and Harald. *Mathematical Methods of Statistics.* Princeton Univ. Press, (1946). ISBN: 0-691-08004-6.

[24] Rao and Calyampudi Radakrishna. "Information and the accuracy attainable in the estimation of statistical parameters". In: *Bulletin of the Calcutta Mathematical Society* Vol.37 (1945), pp. 81–89. DOI: `https://mathscinet.ams.org/mathscinet-getitem?mr=0015748`.

[25] Carlos M. Jarque and Anil K. Bera. "A Test for Normality of Observations and Regression Residuals". In: *International Statistical Review / Revue Internationale de Statistique* Vol.55.No.2 (Aug., 1987), pp. 163–172. DOI: `https://doi.org/10.2307/1403192`.

[26] S. S. Shapiro and M. B. Wilk. "An Analysis of Variance Test for Normality (Complete Samples)". In: *Biometrika* Vol.52.No.3/4 (Dec., 1965), pp. 591–611. DOI: https://doi.org/10.2307/2333709.

[27] John H. Cochrane. *Asset Pricing.* Princeton Univ. Press, (18 Feb. 2005). ISBN: 978-0691121376.

[28] David A. Dickey and Wayne A. Fuller. "Distribution of the Estimators for Autoregressive Time Series With a Unit Root". In: *Journal of the American Statistical Association* Vol.74.No.366 (Jun., 1979), pp. 427–431. DOI: https://doi.org/10.2307/2286348.

[29] P. C. B. Phillips and P. Perron. "Testing for a Unit Root in Time Series Regression". In: *Biometrika* Vol.75.No.2 (1988), pp. 335–346. DOI: https://doi.org/10.1093%2Fbiomet%2F75.2.335.

[30] D. Kwiatkowski et al. "Testing the null hypothesis of stationarity against the alternative of a unit root". In: *Journal of Econometrics* Vol.54.No.1-3 (1992), pp. 159–178. DOI: https://doi.org/10.1016%2F0304-4076%2892%2990104-Y.

[31] H Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* Vol.19.No.6 (1974), pp. 716–723. DOI: https://doi.org/10.1109%2FTAC.1974.1100705.

[32] Gideon E. Schwarz. "Estimating the dimension of a model". In: *Annals of Statistics* Vol.6.No.2 (1978), pp. 461–464. DOI: https://doi.org/10.1214%2Faos%2F1176344136.

[33] G. M. Ljung and G. E. P. Box. "On a Measure of a Lack of Fit in Time Series Models". In: *Biometrika* Vol.65.No.2 (1978), pp. 297–303. DOI: https://doi.org/10.1093%2Fbiomet%2F65.2.297.

[34] J. Durbin and G. S. Watson. "Testing for Serial Correlation in Least Squares Regression, II". In: *Biometrika* Vol.38.No.1-2 (1951), pp. 159–179. DOI: https://doi.org/10.1093%2Fbiomet%2F38.1-2.159.

[35] Karl Pearson. "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *Philosophical Magazine* Vol.50.No.302 (1900), pp. 157–175. DOI: https://doi.org/10.1093%2Fbiomet%2F65.2.297.

[36] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biology* Vol.5.No.4 (1943), pp. 115–113. DOI: https://doi.org/10.1007/BF02478259.

[37] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological review* Vol.65.No.6 (1958), pp. 386–408. DOI: https://psycnet.apa.org/doi/10.1037/h0042519.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* Ed. by David E. Rumelhart and James L. Mcclelland. MIT Press, 1986. Chap. 8, pp. 318–362.

[39] Claude Lemaréchal. "Cauchy and the Gradient Method". In: *Documenta Mathematica* Extra Volume: Optimization Stories (2012), pp. 251–254.

[40] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[41] Xavier Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics* Vol.9 (2010), pp. 249–256. URL: `http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf`.

[42] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: `http://arxiv.org/abs/1412.6980`.

[43] Boris Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Ussr Computational Mathematics and Mathematical Physics* 4 (Dec. 1964), pp. 1–17. DOI: `10.1016/0041-5553(64)90137-5`.

[44] Geoffrey Hinton and Tijmen Tieleman. "Coursera class on neural networks". In: lecture 6 (2012), slide 29. URL: `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

[45] Jeffrey L. Elman. "Finding Structure in Time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: `https://doi.org/10.1207/s15516709cog1402\_1`.

[46] Michael I. Jordan. "Chapter 25 - Serial Order: A Parallel Distributed Processing Approach". In: *Neural-Network Models of Cognition.* Ed. by John W. Donahoe and Vivian Packard Dorsel. Vol. 121. Advances in Psychology. North-Holland, 1997, pp. 471–495. DOI: `https://doi.org/10.1016/S0166-4115(97)80111-2`.

[47] Aurélien Géron. *Hands-On Machine Learning With Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems.* Oreilly  Associates Inc, (11 October 2019). ISBN: 978-1492032649.

[48] P. J. Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: `10.1109/5.58337`.

[49] Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance: From Theory to Practice.* Springer, (1 July 2020).