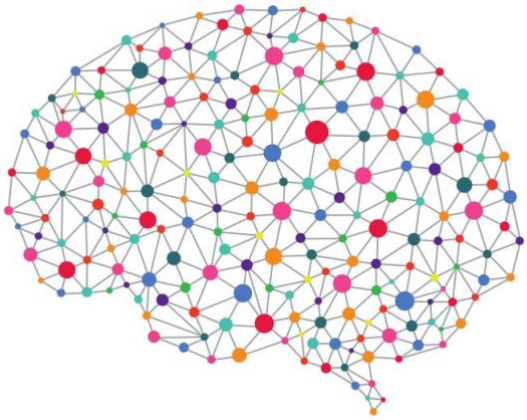
COLEGIO UNIVERSITARIO DE ESTUDIOS FINANCIEROS

DEGREE IN BUSINESS ADMINISTRATION

GRADUATE THESIS: ANALYSIS OF FINANCIAL DATA

INTRODUCTION TO A TRADING STRATEGY FOR CRYPTOCURRENCIES

A MACHINE LEARNING APPROACH



Author: Vaca Serrano, Alejandro

Tutor: Queralt de las Matas, Ricardo

Madrid, June 2018

TABLE OF CONTENTS

1. INTRODUCTION

Since the appearance of Bitcoin in 2008, and especially after its listing on exchanges, cryptocurrencies have attracted many investors' attention. This, however, represents a new asset class with features different to those of more traditional assets, therefore, not much work has been developed as to how to design a cryptocurrency-based portfolio. Nevertheless, in the recent years, many researchers have explored different models for forecasting bitcoin prices, as well as trading strategies for this new asset class.

Being it a revolutionary way of exchanging information, some authors regard Blockchain as the Internet of Value, as opposed to the Internet of Information. This could mean a shift in how humanity manages their assets, how societies are organized, the forms and structures that enterprises take, and many other aspects in our everyday life. It is also probably the first time in history that a technology represents an ideology, that is, Blockchain is profoundly inspired by libertarian and freedom ideas.

Moreover, it encourages safer and more transparent systems; taking out the necessity to trust in one another ironically has the opposite effect: it produces trust among the network participants, given that it is ensured by Blockchain technology, and the information contained in it can be empirically verified, and it is almost impossible to alter it. Opacity and deception are not as possible as with the Internet of Information, in which data is centralized and controlled by monopolistic economic agents, and which therefore damages the most vulnerable segments of the population. The increasing concern by regulators in this respect is a proof of this, and it can be cushioned through Blockchain technology implementation.

Therefore, for these and other reasons that will become clearer along this project's lines, we can regard this type of investment as socially beneficial and speculative, given that a correct implementation of Blockchain technology can lead to achieve social improvements, while it is still a highly volatile, immature asset class.

In this project, the main features of Blockchain, the technology behind cryptocurrencies, are explored, together with few business and real-life applications examples that help to foresee how it can be implemented to solve existing problems, the main one of them being the dependence on trust, which drives many business and investment decisions, and which is crucial for having a healthy and prosperous economic and social space, with a special mention to intermediary companies. That brief analysis provides an overview of the idiosyncratic reasons for engaging in the investment activity with these assets.

Then, the fundamentals of Blockchain technology are explored, to show how a crypto assets portfolio can be created, using various authors' ideas, taking the best from each for the task at hand and putting them together. For that, the first step is to classify cryptocurrencies, which is done first by their underlying technology and objective, and then a hierarchical cluster analysis is performed to check if the intuition that cryptocurrencies with similar objectives behave similarly across time holds true or not.

Lastly, and most importantly, the project focuses on developing a trading strategy for bitcoin and ether (the token used by Ethereum Blockchain). For that end, machine learning and deep learning approaches are used to forecast these cryptocurrencies' closing prices one day ahead. These models are compared in terms of accuracy, and then in terms of the cumulative return that a simple trading strategy based on them would have achieved, especially comparing them against the Buy & Hold strategy from 2017-10-07 to 2018-03-22. This backtesting shows some insights about the desirability to use each of the algorithms tested in a real-life setting. Their limitations are also assessed, as well as the future work that should be developed to build an even more successful and robust trading strategy based on Machine Learning or Deep Learning.

2. UNDERSTANDING THE TECHNOLOGY BEHIND CRYPTOCURRENCIES

2.1. WHAT IS BLOCKCHAIN?

In 2008, after the bankruptcy of Lehman Brothers, Bitcoin surged. It was the first digital cryptocurrency working with Blockchain technology. It was aimed to solve many of the issues presented by previous digital currencies, as DigiCash (Antonopoulos, 2017) (Banking on Bitcoin, 2016); the main one of them being the problem of double spending (Nakamoto, 2008).

After Bitcoin, many other cryptocurrencies were created, using the same Blockchain technology, with new features that improved the network performance in various ways.

Blockchain is a P2P protocol that uses encryption techniques developed in the second half of the last century aimed at building a network for peers to exchange value with no middleman involved. It works as a decentralized major ledger shared by all members of the network, in which all transactions are recorded and which is almost immutable, that

is, it is nearly impossible to modify the information contained in the ledger (Preukschat et al, 2017).

The value unit of a Blockchain is called token (Preukschat et al., 2017). The Bitcoin protocol uses the bitcoin token, with lower letters. The so-called cryptocurrencies are the tokens used in a Blockchain to make transactions and provide incentives for the safeguarding and recordkeeping of the major ledger.

Encrypted transactions in the network are recorded in *blocks*, which have a certain capacity; these blocks are also encrypted using hash functions, in the case of Bitcoin the method used is the SHA-256. When a block is full, a new block must be created to keep storing transactions; it needs to refer to the previous blocks following the structure of a Merkle tree, therefore, a valid *nonce* must be found such that the hash of the *nonce* plus the previous block, given certain constraints, includes all the information contained in the previous blocks (Tapscott and Tapscott, 2016) (Antonopoulos, 2017).

This entails solving a very difficult mathematical and computational problem, which the so-called *miners* need to get right to create the next block. *Miners* are individuals or groups committing their energy and computational power to the maintenance of the network, that is, they are constantly trying to find a valid *nonce* to create the next block, for which they receive a certain amount of bitcoin as an incentive. In doing so, transactions are verified, so that records in the major ledger are consistent. This process determines how tokens are created and issued.

This amount of bitcoin (incentives for block solving) is divided by half every 210,000 blocks, approaching a limit, so that the maximum amount of bitcoin in circulation is known: it will be 21 Million Bitcoin by 2140 (Tapscott and Tapscott, 2016). This process of bitcoin creation (next block generation) happens on average every 10 minutes, following a Poisson distribution. This means that the bitcoin supply along time is known and cannot be manipulated by a central authority, as it can happen today with fiat currencies or companies' shares. This leads to the conclusion that the determinants of its price will depend largely on the demand side.

For a more complete and detailed overview of Blockchain technology, see Antonopoulos (2017), Fantazzini et al. (2016), Tapscott and Tapscott (2016) and Preukschat et al. (2017).

2.2. REASONS FOR INVESTING ON BLOCKCHAIN

Blockchain is aimed at solving many of the problems encountered in the current financial system, and it is almost a serendipity that it was released shortly after the worst catastrophe in the banking sector. They eliminate the need of a middleman, providing faster and safer transactions with an almost null cost (Antonopoulos, 2017) (Preukschat et al., 2017). This technology can have many different uses, although most of its applications released up to date have been in the financial sector. As it works as an (almost) immutable ledger, it can serve us to record valuable information of any kind, therefore it has the potential to transform how our economy and enterprises work and how we exchange value. As Preukschat et al. (2017) and Tapscott and Tapscott (2016) argue, the release of this technology is the transit from the Internet of Information to the Internet of Value.

The main reason for investing in Blockchain, as a general concept, is therefore that it has some properties that make it a likely and desirable substitute of some current institutions and practices. It can enforce the commercial relationships without the intervention of the legal system, with all the time and money savings that this provides, through Smart Contracts[1] (Tapscott and Tapscott, 2016) (Preukschat et al., 2017). Moreover, it is a global record with no central authority controlling it, and due to its characteristics, this is a more robust method to keep private data safe, while having the advantages we enjoy nowadays (Antonopoulos, 2017).

There are plenty of examples in numerous industries showing how Blockchain can add value compared to current practices. In the energy sector, it provides the opportunity to build a decentralized system in which, instead of having few power generation plans, often distanced from large consumption centers, there would be many small power generation plants, most of them for self-consumption. When a self-consumer has extra production, it is released to the network in exchange for tokens; the opposite happens when they have production deficit. Companies like Powerpeers, Consensys with LO3 and Grid Singularity are already working on models like this (Preukschat et al., 2017).

This reduces the energy loss in transportation and is a more inclusive approach, letting consumers benefit from their use of resources; therefore, such an energy generation model

[1] For simplicity, a Smart Contract is a collection of code that specifies an agreement between two or more parties, in which every possible outcome from it is agreed before signing the contract; they execute themselves given certain previously agreed conditions (Antonopoulos, 2017). This eliminates the need of a legal middleman to enforce a commercial relationship.

can create value for consumers and provide a greener energy industry in which resources are better managed, advocating for circular economy.

In the case of global payments, Blockchain usage offers a qualitatively different service compared to traditional companies in this sector as Western Union. For immigrants who send money to their families in their homeland, having the opportunity to send funds from their smartphone (as compared to the long queues of Western Union offices), paying less than 1% network fee (Western Union charges around 9% fees) (Tapscott and Tapscott, 2016) (Preukschat et al., 2017) and knowing their funds will reach their destiny in minutes (as compared to days with traditional money transfer companies) has a huge value (Tapscott and Tapscott, 2016).

Furthermore, 38% of the World's population do not own a bank account (Preukschat et al., 2017). Many of them live in developing countries, where they encounter the added problem of their governments' monetary policies, in most cases provoking a huge inflation. Blockchain brings the possibility to these members of society to be included in the global financial sector under a known and automated value units' supply. This can boost the emerging economies' growth, providing their members with a platform for them to have access to financial services adapted to their needs. Given the huge unexploited potential of those countries, the emergence of this technology can help to reduce the increasing spread between developed and developing countries, most accentuated during the digital age (Brynjolfsson and McAffee, 2014). It is therefore reasonable that investors take positions in such assets, as they can generate exceptional value, also for an unserved and important in size part of the population.

Other industries have already started testing the potential benefits of Blockchain. It is the case of intellectual property. As Brynjolfsson and McAffee (2014) point out, artists and other content creators have seen their rights violated during the digital era. However, this would not be possible in a Blockchain ecosystem, as everything is recorded in the network and the content is not duplicable due to the mathematical connection between transactions (in this case, think of a song or another piece of art as a transaction) (Preukschat et al., 2017) (Tapscott and Tapscott, 2016). For using another person's content, consumers would have to make micropayments according to Smart Contracts. Closely related with this Blockchain aspect, the traceability and immutability of information may help to get rid of fake news, which, as shown by the Cambridge Analytica's US elections manipulation (Cadwalladr and Graham-Harrison, 2018), poses a problem for people's freedom.

The possibility of making micropayments is also greater with Blockchain, as its logistic is very flexible and cheap, compared to banks, and Blockchain's tokens are typically highly divisible (Kuo Chuen, 2015). Many consumers would pay a thousandth of a dollar for a song, but currently they do not have the possibility to do so (Preukschat et al., 2017). Tapscott and Tapscott (2016) and Preukschat et al. (2017) provide other examples and benefits of Blockchain usage in the health sector, the gaming industry, the food sector and insurance companies, among many others. What all the uses of this technology have in common is that they provide a safer data-keeping method, which is also cheaper and much more inclusive, allowing all the members of the population to have a more active and important role in the global economy. Some other relevant expected contributions are the increase of political and public accounts movements transparency, which, in countries like Spain this can be considered crucial, and more focus on data traceability and control by consumers.

It is therefore clear that Blockchain already adds much value to the current economic system, and given that it is only on its early stage, it is expectable that it will become more determinant in the coming years. Enterprises or institutions leveraging Blockchain technology do not necessarily possess their own Blockchain and tokens; the value added for the mentioned benefits is also captured in the Blockchain's tokens they use for that end.

Furthermore, as Preukschat et al. (2017) and Kuo Chuen (2015) point out, one difference between the Internet of Information and the Internet of Value is that in the latter the protocol layer is the one that captures most value. The implications of this for investing are that while you cannot invest in the TCP/IP protocol, but on the applications built on top of it, as Amazon or Google, you can indeed invest in the Ethereum protocol, on top of which most ICO's are created. There are other protocols of this type, like EOS or Lisk. You need to buy first some of these protocol Blockchains' tokens for investing on the applications (companies) built on top of them, which creates a natural demand for the fat protocol token.

As explained earlier, companies using Blockchain can create immense value for consumers; most of these are built on top of protocols like Ethereum, from which it is concluded that investing on such protocols' tokens can be highly profitable and provides diversification, as even if some of these applications fail, the overall value of the protocol is not too dependent on an individual application's performance. We will refer to those cryptocurrencies as protocol tokens or protocol cryptocurrencies.

Though, some of these companies built on top of a Blockchain protocol like the ones described above generate benefits for consumers and therefore are taking advantage of the release of Blockchain for creating economic value. They may be regarded as a highly speculative investment, as they are the newest form of crypto assets with future projection; however, they have some desirable properties that make them good candidates for being part of a crypto assets portfolio. These will be mentioned under the name speculative tokens or speculative cryptocurrencies.

We can divide crypto assets into more groups apart from protocol tokens and speculative tokens, following the classification by the advisor, angel investor and finance writer Ryan (2017), who also provides metrics on the potential of each cryptocurrency. The third type of cryptocurrencies are the major or core ones. It is composed of those that have been more time in the market, and therefore are the ones most investors focus on, as well as the currently most used. Ether and bitcoin are the main tokens in this respect. These tokens are the base monetary units in most exchanges, meaning that for purchasing other cryptocurrencies you need first to have enough of these.

Anonymity cryptocurrencies are focused on improving Blockchain technology for digital invisibility, which was one of the reasons behind the creation of the technology in the first place. In the current Big Data era, in which consumers' data privacy is highly violated (Brynjolfsson and McAffee, 2014), such cryptocurrencies can become popular among the population.

The group of "zombie currencies", formed by forks of major coins or copies of others adding no relevant technological innovations or unique value proposition (Ryan, 2017), are not considered.

The last group of cryptocurrencies is the so-called platform tokens. These are centralized platforms, like Ripple or NEO, which use Blockchain technology to solve different problems in the economy; in the case of Ripple, for example, xrp tokens are used to leverage the scalability and efficiency of the Ripple platform to execute global payments much faster than with traditional methods (Preukschat et al, 2017).

As a summary, Blockchain technology provides a faster, cheaper, safer method for exchanging value between peers, giving more access to credit to a wider part of the population. It is mainly decentralized, which eliminates the need for trust, and hacker attacks to the whole network, contrary to the current centralized system, are almost impossible. It encourages more transparency while increasing consumers' protection, who get back control over their data, which they can decide to monetize. Through

traceability, fake news negative effects can be eliminated, and it empowers the use of immutable user rates, which incentivize better behavior. Moreover, as Tapscott and Tapscott (2016) and Preukschat et al. (2017) argue, Blockchain implementation is crucial for a safe and robust Internet of Things (IoT). Finally, there are different types depending on the needs of their applications.

3. DESIGNING A PORTFOLIO IN CRYPTO ASSETS

3.1. THEORETICAL FRAMEWORK REVIEW AND THE DEGREE OF APPLICATION TO THE CRYPTOCURRENCY MARKET

Traditional investment and portfolio management authors like Grinold and Kahn (1999) and Graham (2006) do not provide a basis for investment on this type of asset, as they are very new. For this reason, the constraints for the design of a trading strategy on cryptocurrencies investment are obvious, as there is little theoretical background for that. However, there are some common notions regarding how markets as a general concept work, and although the techniques used for shares, derivatives or bonds are not applicable, there are others that can be adjusted for this end.

Section 1.2. discussed the main reasons for investing on Blockchain projects, as well as the different benefits each cryptocurrency category brings to the market. This step of observation and qualitative assessment for forming beliefs about an asset type is the first consideration in portfolio selection by Markowitz (1952). The next step would be to design a portfolio such that the returns are maximized while volatility is minimized.

This approach has two main limitations. The first one is that this strategy would not take advantage of or protect against extreme unexpected events in the market (sharp losses or gains in a matter of hours), which are more likely in such a new market with no clear regulation yet, as in extreme volatility situations most cryptocurrencies move in the same direction (Preukschat et al., 2017), see figure 1. The second limitation is that this approach would reduce opportunistic investment on newly created Blockchain tokens, as we would not have enough historical data to compute the covariance between those and the other cryptocurrencies, as needed to follow the method described in Markowitz (1952) and implemented by Bocconi Students Investment Club (2017).

As a matter of fact, it was not expected that all cryptocurrencies experienced a huge rally between December 2017 and the beginning of January 2018; it was also not expected that

within days, half of that market value was lost. This provides an insight into the nature of the cryptocurrency market: it is extremely volatile and it is highly dependent on the news and rumors on the Internet. Being them digital assets, it is not surprising that their markets depend much on the related content in it (Kuo Chuen, 2015).

News and rumors ultimately provoke fear or excitement in the markets, closely related to the first of the animal spirits described by Shiller and Akerlof (2010), which is confidence and its derivatives; and the famous 'irrational exhuberance'. Thaler and Sunstein (2008) and Kahneman (2012) analyze the impact of loss and risk aversion on the markets, as well as investors' decision making process, which is very prompted to biases and heuristics; this is exacerbated in immature markets with more uncertainty and full of unexperienced investors (Tversky and Kahneman, 1974) (Tversky and Kahneman, 1979). This brings much volatility and confusion in the cryptocurrency market, in which restrictive regulation rumors or exchange platforms' hack attacks news make prices dip, as it happened in mid-January (Williams-Grut, 2018) (Pollock, 2018). On the other hand, good news or rumors as the listing of Bitcoin Futures in the Chicago Mercantile Exchange (CME) and Chicago Board Options Exchange (CBOE) cause prices to go up rapidly, as in December 2017.

As a summary, the main characteristics of the Blockchain market affecting the portfolio choice are the following. It is an immature market with extreme volatility driven by investors' sentiment, so very risk-averse investors who want to avoid volatility should not enter the market by the moment. It is constantly growing, as it is on its early stage and new companies are entering every week, therefore for many projects there is no historical data to assess possible future performance.

There are very different cryptos regarding use, technology, supply, objective and other characteristics; so, generalizations cannot be made, which limits the possibility of comparing them. They can be divided into categories as Ryan (2017), as did in the previous section, but even so, many of them will fall in more than one category at the same time. Finally, the most important risk for the industry is regulation (Sharma, 2017), which affects all cryptocurrencies (not equally though), as it is imposed on the whole cryptocurrency market.

Taleb (2007) explains the concept of asymmetric investment, that is, an investment choice for which we know the potential loss and it is very low, but for which we do not know the potential gain. It is a common-sense choice to select assets for which we can only lose what we invest, being this a relatively very small quantity, but with a huge potential

growth. For this reason, with the objective to take advantage of randomness instead of being weak against it (Taleb, 2004), it is reasonable to include in the portfolio small tokens from new companies that are not very well-known to the public yet, but which have strong developer support and that have a clear value proposition.

For this reason, a small portion of the portfolio should be dedicated to speculative cryptocurrencies, as they are very cheap, so with few dollars lots of them can be bought, therefore keeping the potential loss low, while taking advantage of sudden sharp prices increases, being them boosted by pure speculation or a demand increase for the company's products or services, buyable with their tokens. Taleb (2012) would call this an 'antifragile investment', as randomness and especially black-swan-like behavior can only significantly increase the value of the wallet, while the potential drop is kept at minimum.

The approach followed is therefore a mix of Taleb's barbell strategy (McFarlane, 2018), the diversification classification and potential evaluation by Ryan (2017), the basics of Markowitz's return-volatility relationship (Markowitz, 1952) and the method developed by Kuo Chuen (2015), which takes into consideration the community support, market liquidity as measured in the volume exchanged of each cryptocurrency in the major exchanges and developer activity. For a more detailed explanation on the components of each of these metrics, see Kuo Chuen (2015).

3.2. BUILDING THE PORTFOLIO

As mentioned before, ether and bitcoin are the most important of the major cryptocurrencies, not only for their early mover advantage in their respective submarkets or because they have the largest market capitalization, but because most transactions in exchanges are done with these. They are the most widely adopted and best known, therefore 50% of the portfolio will be initially composed of these cryptocurrencies; 5% will be used for protocol cryptocurrencies (considering that in the 50% of core cryptocurrencies ether, the most important protocol token, is included, a 5% of other protocol cryptocurrencies is enough), 20% for platform cryptocurrencies as well as for anonymity cryptocurrencies, and finally 5% for speculative tokens. The latter are are left for future work and not addressed in this project.

Figure 1 is composed of the daily log returns' correlations of all the cryptocurrencies under consideration[2] from August 08th, 2017 to February 17th, 2018[3]. It is noticeable that the only cryptocurrency (weakly) negatively correlated with the rest of them is maid (Maidsafe). The rest of them are positively correlated with each other, thus proving the previous statement that the Markowitz (1952) method for diversification is not feasible in this market, although its logic can be partly applied: we should select, whenever possible, cryptocurrencies that are as weakly correlated as possible. This figure provides an insight into the collective behavior of the cryptocurrencies under consideration for portfolio selection.

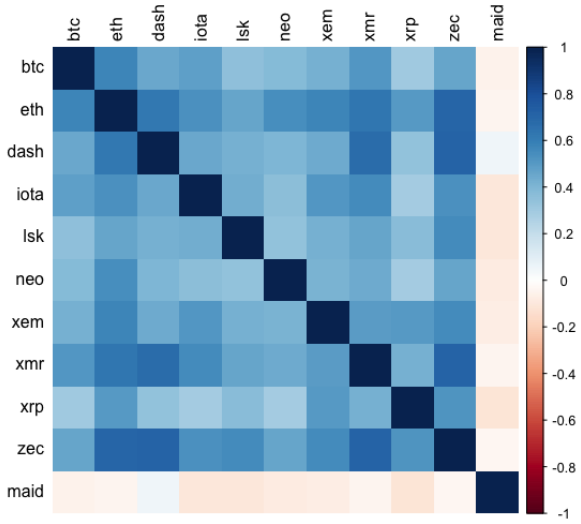


Figure 1. Correlation Matrix of Cryptocurrencies' Log Returns.

Data Source: Cryptocompare (2018).

Figure 2 is a hierarchical cluster of the cryptocurrencies under consideration, using the techniques developed by Montero and Vilar (2014) for performing clustering on time series data, in this case based only on the log returns of the time series for the aforementioned period, very similar to the work by Collier (2017) with US stocks. The full R code for this section can be found in Annex 1.

[2] For core cryptocurrencies, btc and eth (main protocol coin, it could be part of that group but due to its relevance it is worth including it in 'major cryptos' group) were selected. For protocol currencies, iota, lsk (Lisk) and maid (Maidsafe) were selected, partly based on the investment strategy by Ryan (2017). The anonymity currencies under consideration are zec and xmr; finally, xem (NEM), xrp (Ripple), neo and dash (which could perfectly be in anonymity coins' group) were the platform currencies selected.

[3] We use this time frame because August 08th, 2017 is the first day there is data from NEO, the newest cryptocurrency of the ones considered.

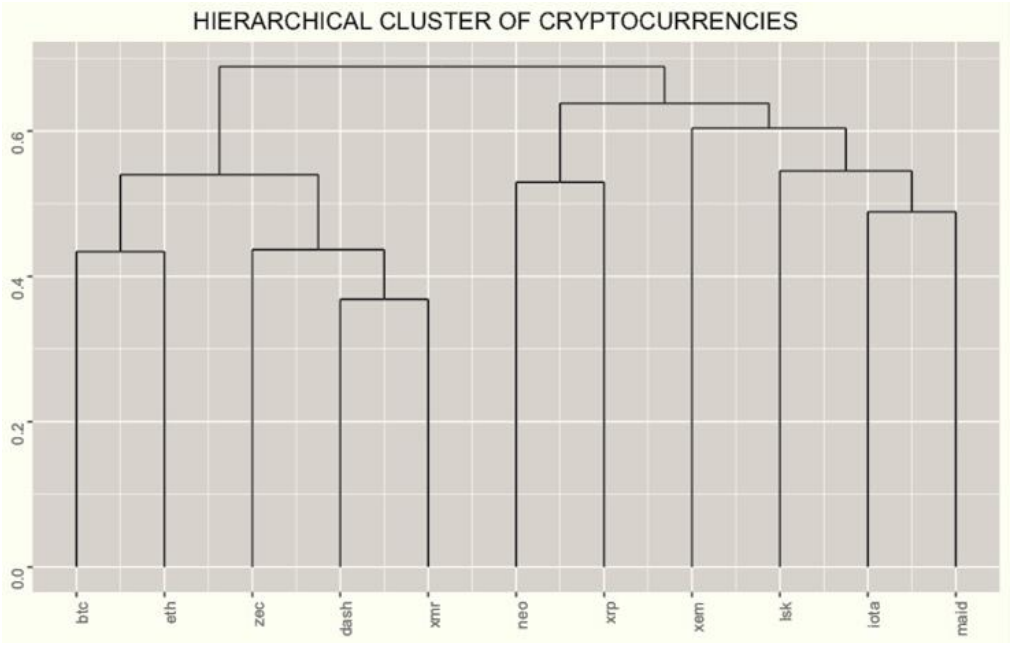


Figure 2. Hierarchical Cluster of Cryptocurrencies Under Consideration.

Data Source: Cryptocompare (2018).

It is noticeable that the classification proposed almost perfectly fits with the one by the hierarchical cluster. This suggests that cryptocurrencies' time series from the same group behave similarly, and that the groups selected are adequate. The cluster brings the possibility to re-adjust the groups before continuing the analysis: dash, as mentioned before, is used mainly as an anonymity cryptocurrency, and it is clear by the graph that it falls under the same roof as zec and xmr, therefore it will be treated as an anonymity cryptocurrency instead of a platform one.

The method used for building the cluster classification was the "ACF" in the TScluster package (Montero and Vilar, 2014); which means that cryptocurrencies in the same cluster behave similarly in terms of their autocorrelation functions (Tsay, 2013). This function can serve as a descriptor of each time series' behavior, thus distributing the portfolio weights among all categories is supposed to have a diversification effect, as the funds are allocated among groups of tokens that behave differently along time, with the limitation that correlation among them is still high.

The tables 5 to 8 in the Annex 7 are composed of the variables considered for portfolio selection for the cryptocurrencies in each group. There is one table per group. The variable 'Ryan (2017) overall grade' refers to the mean of the grades Ryan (2017) gives to each cryptocurrency based on Product/Function, Adoption, Technology, Aligned Incentives/Governance and Market Opportunity. The variables liquidity, community

support and developer activity are taken directly from www.coingecko.com[4] (Coingecko, 2018), 'tot_perc_ret' is the percentage change in the close price from the beginning to the end of the period; 'mean_daily_ret' refers to the mean of the daily log returns of the period, 'mean_daily_vol' is the mean of the daily volatility, measured as 'high' (maximum price of the day) minus 'low' (minimum value of the day).

However, to put this number in context and adjust for the magnitude of each time series, 'mean_daily_perc_vol' was created, which is the average daily volatility divided by the average daily close price. Moreover, as the objective is to penalize too much volatility, the inverse of this number was taken, captured in '1/mean_daily_perc_vol'.

The variable 'av_google_interest' is the mean of the relative percentage interest in each cryptocurrency in the last 12 months, as compared to the rest of the ones forming each group. It was furtherly transformed to 'av_google_interest*10', to have all the variables in the same scale. This transformation was also applied to '1/time', a variable that penalizes for too much time in the market, as with time assets are expected to stabilize more and therefore they are not as good investment opportunity for the future as well performing assets with shorter life (Preukschat et al., 2017).

Finally, the overall weight of each corresponding cryptocurrency is calculated as follows:

$$weight_k = weight_j \cdot \left[\frac{\frac{\sum_{i=1}^{n_k} x_{ik}}{n_k}}{\sum_{k=1}^{k_j} \frac{\sum_{i=1}^{n_k} x_{ik}}{n_k}} \right]$$

Equation 1. Portfolio Weights Calculation.

Source: Author.

for k cryptocurrencies, j cryptocurrency groups mentioned above, x_i are the grades (x) obtained in each variable i[5] for the cryptocurrency for which the weight is being computed; n_k is the number of variables that each cryptocurrency k has. $\sum weight_k = 1$; $\sum weight_j = 1$. Table 1 provides the result obtained after applying this formula.

[4] Coingecko data, as well as google trends data, was taken between February 18th, 2018 and February 22th, 2018.

[5] The final variables for portfolio selection were the following: Ryan (2017) overall grade (average of each grade), community support, developer activity, liquidity (all three in a 1-10 scale instead of percentage), 'tot_perc_ret', 'avg_google_interest*10', '1/mean_daily_perc_vol', '(1/time)*10'. Regarding the last variable, time was measured in years, for which the number of days since the first appearance of the crypto in www.cryptocompare.com was divided by 365.

Crypto	% portfolio
BTC	25,6%
ETH	24,4%
IOTA	1,6%
LSK	1,8%
MAID	1,5%
XMR	6,5%
ZEC	6,4%
DASH	7,0%
XEM	5,9%
XRP	5,8%
NEO	8,3%
SPECULATIVE	5%

Table 1. Theoretical Portfolio

Data Sources: Cryptocompare (2018), Google Trends (2018), Ryan (2017), Coingecko (2018).

It is noticeable that btc and ether have a huge weight in the portfolio, which makes sense since they are the most known players in the market, with them together accounting for 57.92% of cryptocurrency market capitalization as of February 20th, 2018 (Coinmarketcap, 2018). This motivates the focus on these two, exacerbated by them being the main base cryptocurrencies for the rest of trades in most exchanges. Due to this, the next section focuses on the design of predictive models for these two cryptocurrencies.

4. DEVELOPING PREDICTIVE MODELS FOR BITCOIN AND ETHEREUM

4.1. LITERATURE REVIEW (PREVIOUS WORK)

Kristofuek (2015) uses wavelet coherence analysis to detect the determinants of bitcoin price; however, his framework is not useful for forecasting. Tarnopolski (2017) uses a Monte Carlo approach with geometrical fractional Brownian motion for forecasting bitcoin price in the long-term. However, given his early 2018 poor prediction, it is concluded that predicting cryptocurrencies in the long term may be an almost unfeasible task, given the immaturity of the market, the volatility and the rest of the market's features previously explained.

Georgoula, et al. (2015) pursue the same objective as Kristofuek (2015) with time series and sentiment analysis. While they do not provide a forecasting structure, it brings some insights into the variables that can cause the variation in bitcoin prices, as the sentiment in social media (news and rumors previously discussed), interest (taken from google searches and Wikipedia searches) and technological state of the cryptocurrency (hash rate). Nevertheless, their approach has another limitation, apart from the preceding one: retrieving daily data for a sufficiently long period from Twitter, google and Wikipedia has become an almost unfeasible task, due to the APIs limitations and the elimination of some of this data from the source used by these researches for Wikipedia and Google data[6].

In this direction, Zhu, Dickinson and Li (2017), conclude that bitcoin does not behave as a currency, and should more likely be considered a speculative asset (a similar guess to the one provided in previous sections); Yermack (2013) claims something very similar. Zhu, Dickinson and Li (2017) affirm that macroeconomic magnitudes such as gold price and the US dollar index affect bitcoin's price in the long term.

Poyser (2017) got similar results, providing a bunch of macroeconomic variables affecting its price. However, there is a limitation in finding the relationship between variables such as the stock market index or gold price with bitcoin's price daily, as the former variables do not have data for weekends, while the bitcoin market is always active; we could fill the missing days with Friday's close value, but this would be more detrimental than helpful for forecasting purposes. That possibility was already tested, finding that the above intuition holds: filling the missing days is detrimental.

Therefore, the findings of Zhu, Dickinson and Li (2017) and Poyser (2017) are not useful for daily forecasting and trading, although they show that despite being a speculative asset, it is also slightly related with macroeconomic developments, which can be good for its market long term growth, as people may rely on it when looking for an alternative to the traditional markets.

Ciaian, Rajcaniova and Kancs (2015) used an augmented Barro's equation, initially developed for gold (see Barro (1979) for more detail). Their results differ significantly from Zhu, Dickinson and Li (2017) and Poyser (2017) regarding macroeconomic variables, as the only significant one they found dealt with market fundamentals, captured by Barro's equation, and attractiveness for investors. Chu, Nadarajah and Chan (2015)

[6] www.bitcoinpulse.com

contribute with some insights into the bitcoin price behavior, claiming that the generalized hyperbolic distribution is the one, of all known parametric distributions, that fits best bitcoin returns.

Fantazzini et al. (2016) make a review of the main methods developed until the date of their writing in modelling bitcoin prices. They also give a very clear explanation of how bitcoin works, as well as of the main statistics regarding the cryptocurrency. In general, their work overviews the main contributions to the field; however, it is not specialized on forecasting and does not have much content on this. Nevertheless, their findings in page 18 about an increase in the computational power used in the network having a positive impact on price, as well as the inverse relationship between the number of coins found (mined) per minute and price, contribute to forming an idea of the possible variables affecting bitcoin's price variation.

Shah and Zhang (2014) developed a very successful method for bitcoin price forecasting, using Bayesian regression together with the latent source model, therefore approaching the prediction task as a binary classification problem. Nevertheless, the dataset used, comprised of months of 10-seconds intervals, as well as the computational power required to develop that methodology, are out of my reach. Moreover, they use high-frequency ask and bid data I have not access to by the moment.

Several attempts have been made to either predict or trade on models using sentiment analysis (mainly from Twitter data) as the explanatory variable. Kim et al. (2016) try to predict fluctuations in bitcoin prices based on user comments and replies, using a mix of web scraping, sentiment analysis and machine learning. Their results show that sentiment analysis methods can be very effective to predict bitcoin prices, supporting the previous claims on rumors and news effects on this market. Stenqvist and Lönnö (2017) use Twitter sentiment analysis to perform a similar task, while Colianni, Rosales and Signorotti (2015) use it to design a trading strategy on bitcoin.

Khandelwal, Adhikari and Verma (2015) use an ARIMA model and an artificial neural networks model based on Discrete Wavelet Transform Decomposition, a similar approach to the one used by Alphamacro (2018) for designing a trading strategy for the SP500, improving the ARIMA model performance by adding wavelet transforms and by Veronin and Partanen (2013) for energy prices forecasting. However, the area of study in which wavelets are covered is very distant from this project, and given the lack of resources on code for using wavelets in time series forecasting, they will not be applied

here. Nevertheless, both authors claim that their inclusion significantly improves forecasting accuracy, therefore this area of study should be expanded.

MacDonell (2014) use an ARMA model to predict daily bitcoin prices augmented with a log-periodic power law for predicting crashes; the first part of his methodology is the most relevant for the aim of this work. Hencic and Gourieoux (2015), on the other hand, use a non-causal autoregressive model. It is evident that autoregressive models are a typical approach when confronted with time series forecasting.

One of the best attempts to predict a cryptocurrency price accurately is Amjad and Shah's (2016). They use an ARIMA (4,1,4) model along with a Logistic Regression (LR), Random Forest (RF), Empirical Conditional Distribution (EC) and Linear Discriminant Analysis (LDA) to forecast bitcoin prices. They show that despite ARIMA forecasting accuracy (2015: 0.49, 2014: 0.43), all the rest of the models outperform it. EC got 0.64 both in 2015 and 2014, RF got 0.78 in 2015 and 0.69 in 2014. LDA got 0.73 and 0.71 respectively, while LR got 0.70 and 0.69.

The Logistic Regression model was the one with the highest Sharp Ratio in 2015 (3.32), as well as the one attaining most profit (9090.9 compared to 1366.7 for ARIMA). However, notice that they use several months of second by second bid-ask data, a similar limitation to the one encountered with Shah and Zhang's (2014) approach.

McNally (2016) uses a Bayesian optimized recurrent neural network (RNN) and a Long Short-Term Memory neural network (LSTM) for forecasting bitcoin price one day ahead; contrasting these results against those of ARIMA. As expected, both deep learning methods perform better than the econometric one. Another completely different method is the one used by Berlinger et al. (2015). They use technical indicators as RSI and MACD to build a neural network with these indicators as the only input layer, and next day's bitcoin log returns as the output.

Greaves and Au (2015) use network information extracted directly from the Bitcoin's Blockchain to forecast bitcoin prices one hour in advance, using and comparing Linear Regression, Logistic Regression, Support Vector Machine and Artificial Neural Networks, with the latter attaining the best classification accuracy, with a 55%. Hegazy and Mumford (2016) use different classification algorithms with the same purpose, the most profitable being Recurrent Reinforcement Learning (RRL), although the most accurate (higher correct rates) was Boosted Trees.

Madan, Saluja and Zhao (2014) also use a classification approach, attaining a 98.7% accuracy using a binomial generalized linear model. However, using the same variables

as they do, I got a 59% accuracy, as seen in Annex 2. This suggests that their results correspond to a model in which the explanatory variables are in the same moment in time as the explained variable, which is not useful for forecasting. Other possible reason is that as their paper is written in 2014, the bitcoin price changes do not behave like that anymore, but did in the past.

It is remarkable that most of the literature reviewed so far deals only with the modelling or prediction of bitcoin price, therefore for ether there is less previous work to rely on. This was expected since ether is a much younger cryptocurrency with a significantly lower market capitalization, therefore, researchers' interest has been focused on bitcoin as the first, main and most popular cryptocurrency. The most relevant article purely regarding ether price forecast is the one by Chen, Narwal and Schultz (2017); they find that the best model for predicting ether prices one day ahead was an ARIMA model.

Finally, models ARIMA and LSTM are used for forecasting both ether and bitcoin prices one day in advance. Then, their results are compared. The ANN model using technical indicators developed by Berlinger et al. (2015) was replicated[7], finding so disappointing results that these were not even included in this project. A possibility for further research would be to try a similar approach but using LSTM instead; taking the technical indicators in Berlinger et al. (2015) as the input layer and then letting LSTM decide which connections between them are relevant.

4.2. DATA

For the explanatory variables exploration, the dataset in Kaggle (2018) is used, which already contains the variables retrieved from Bitcoin's Blockchain API. For all the ARIMA and ARIMA + GARCH models, data is directly taken from Cryptocompare (2018). On the other hand, for LSTM the data is retrieved from Coinmarketcap (2018) (1) (2) (3) (4). In the case of ARIMA, the daily bitcoin and ether prices from 2015-10-18 to 2017-09-30 and 2017-06-05 to 2017-09-30 respectively (Period 1) are predicted. However, the time frame of interest for this project is from 2017-10-01 to 2018-03-22 (Period 2), which is equal for both cryptocurrencies. This second period is forecasted

[7] This replication for the period of interest for this project can be found in: https://github.com/alexvaca0/TFG18/blob/master/ann_technical.Rmd. It was uploaded to Github but no space is dedicated for it in these lines because it proved not to be reliable; the ANN is not able to capture any of the bitcoin price movements; therefore, it was not even tried on ether. This suggests that the results by Berlinger et al (2015) only hold for the 30 days in 2014 that they predict.

using ARIMA + GARCH (1, 1). Then, using LSTM, prices from 2017-10-07 to 2018-03-22 are forecasted. All prices are in US dollars.

4.3. ARIMA & ARIMA + GARCH (1, 1) MODELS

Before using the ARIMA approach, a little exploration on the possible variables from the Bitcoin network (Blockchain) affecting its price is carried out. Annex 2 contains all the relevant code for this part, including a brief analysis on the impact of each variable on bitcoin price; it also contains an ARIMAX model using the significant variables previously discovered with stepwise selection, and the Binomial GLM previously mentioned. However, these variables do not affect the next day's bitcoin log return, as seen in Annex 2. This brief exploration, despite not useful for forecasting purposes, provides an insight into the mechanics of bitcoin price formation.

As seen in sub section 4.2., there have been many attempts to use autoregressive models to forecast bitcoin prices; in fact, it is one of the most known and used methods for univariate time series prediction (Tsay, 2013). This is due to the characteristics mentioned by Amjad and Shah (2016), confirmed by the ACF and PACF plots in Annex 7 (figures 9 to 12) showing stationarity[8] and mixing: there is significant dependence among values, with much more effect from most recent observations. This seems the perfect scenario for an ARIMA model to fit in. For a detailed explanation of the model see Tsay (2013).

Amjad and Shah (2016) found that the best ARIMA model for the time frame they studied was of order (4,1,4). Though, they use intervals of seconds while the method described here is for daily data, therefore the results are not comparable. To improve forecasting accuracy, I have added a few contributions to the model, using a machine learning approach instead of an econometric one.

First, p and q in the model are not chosen previously, but these are automatically optimized for better fit using recursion over the time series, so that for each point forecast the model is re-estimated using all previous data. To ensure stationarity, d is also automatically calculated, using *auto.arima* function from *forecast* package by Hyndman (2017); it uses the KPSS test (Tsay, 2013) to select the optimal order of first-differences, d, so that the time series are stationary (also D, the seasonal difference order, is selected automatically if needed). Annex 3 contains all the R code used for this section. As seen,

[8] By using the *auto.arima* function, stationarity is ensured.

the model performs an iterative process of going through all previous prices and then forecasting the next one. The window in this first attempt is therefore every day wider.

Unlike Amjad and Shah (2016), no order is selected and therefore the model is not static, but can modify itself with new information. Differing from the typical machine learning procedure, the steps followed are not the conventional (train and test). As seen in the code in Annex 3, the model first trains on some data, and then this training period is extended by 1 every iteration, up to the previous day of the forecasting day; there is not a test period per se, because the objective is not to find the optimal ARIMA order and then using it for forecasting a long period, but letting R change the best ARIMA order with new data, providing a flexible approach, in which the model can easily adapt to new information, for forecasting next day's price[9] (it is expected that the relationship of current prices with previous ones changes throughout time).

This model is used in two different time ranges, to compare its performance given different market events. It was expected that the model worked well under few or moderate volatility, and that is the case, as can be seen in Annex 3. However, it does not forecast so well in more volatile environments.

For this reason, a GARCH (1,1) was added, to be able to capture the cryptocurrency volatility (Berlinger et al, 2015) (Tsay, 2013). The inclusion of this feature in the model gives the additional advantage of capturing the changes in volatility, therefore, the most probable future points between which the bitcoin price will be, are easier assessed. Moreover, as the method from *rugarch* package (Ghalanos, 2018) *ugarchforecast* is used for forecasting, the point estimator also changes. It is noticeable that in both cases, the forecast seems to fit the actual prices series (see figures 3 and 4).

Another contribution is to design a machine learning approach for selecting the optimal window for the model. This can be seen in Annex 3. The output is not displayed due to the huge time and computing power requirements; the result obtained, which is replicable using the provided code, shows that 400 days before the forecasting day are the optimal ones to set.

Figures 3 and 4 below show the predicted and actual values, as well as the upper and lower boundary[10] for the second period using the optimal window and the *ugarchforecast* function, for bitcoin and ether, respectively. It is observable that the actual price almost always falls between the upper and the lower boundary; consequently, this could be used

[9] The test period is, therefore, 1 day every time.

[10] Calculated as point_prediction $\pm$ standard_deviation_predicted

to ensure the robustness of the trading strategy by evaluating not only the point estimator but the most probable range of prices; therefore, deciding how much exposure to take depending on the most probable market direction derived from that interval.

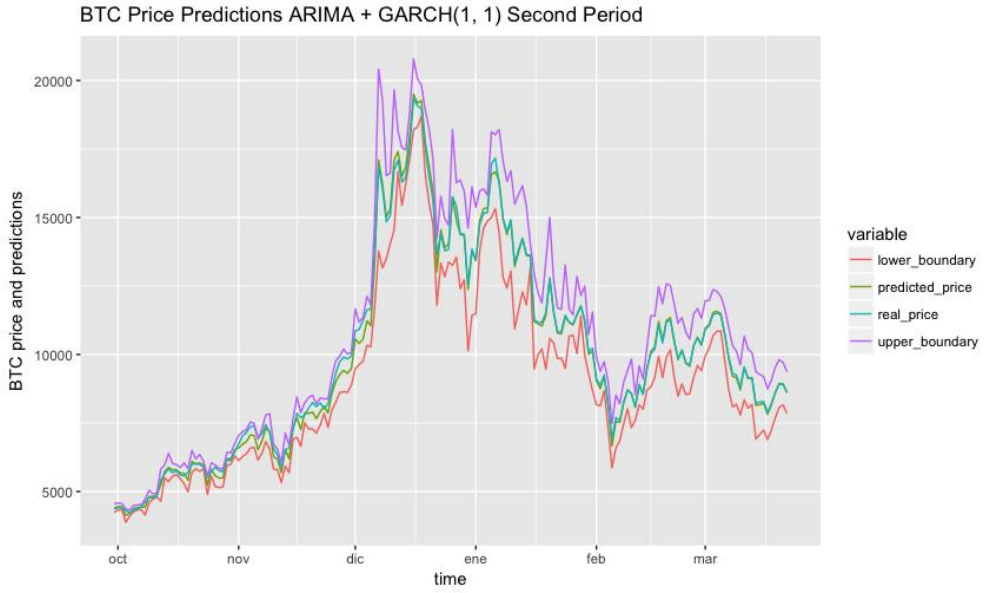


Figure 3. Visualization of bitcoin forecast with ARIMA + GARCH (1, 1).

Data Source: Cryptocompare (2018).

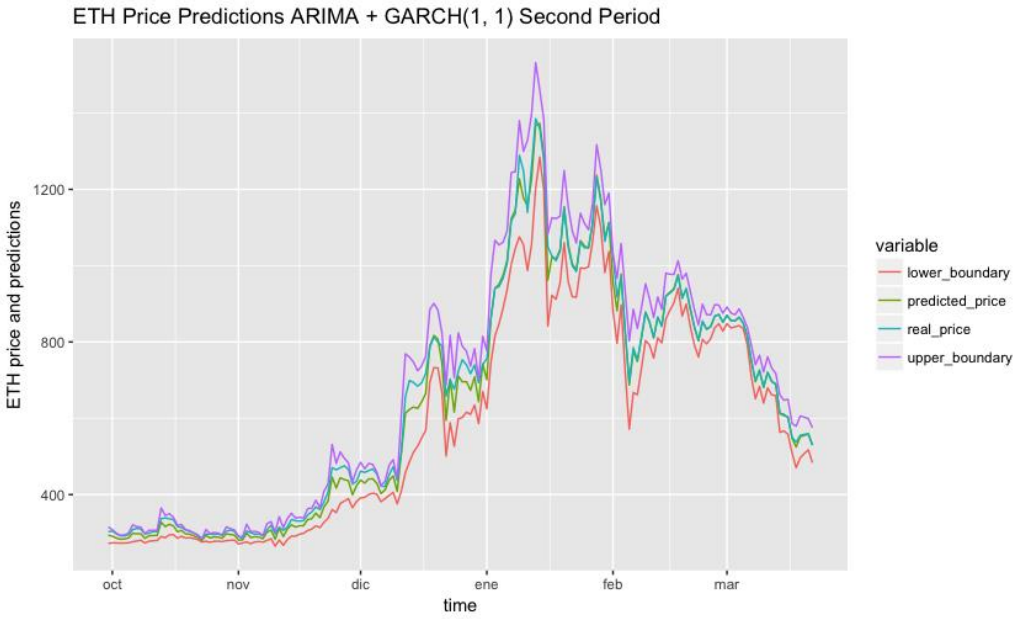


Figure 4. Visualization of ether forecast with ARIMA + GARCH (1, 1).

Data Source: Cryptocompare (2018).

Table 2 contains a summary of the accuracy in each time range described in sub-section 4.2. for bitcoin and ether, respectively, measured as 100 – MAPE.

MODEL AND PERIOD	Bitcoin prediction accuracy	Ether prediction accuracy
ARIMA Period 1.	**97.52%**	**94.45%**
ARIMA Period 2.	**95.07%**	**95.3%**
ARIMA + GARCH (1, 1) Period 2.	**98.40%**	**97.38%**

Table 2. Performance of ARIMA and ARIMA + GARCH (1, 1) models in each period.

Data Source: Cryptocompare (2018).

4.4. LONG SHORT TERM MEMORY (LSTM)

In the last years, deep learning models have become very popular for many different tasks, one of which is time series forecasting. In fact, there have been many attempts to predict stocks and cryptocurrencies' prices using them (Binsumi, 2016) (PiSimo, 2017) (Heinz, 2017) (Aungiers, 2017) (Coinalysis, 2018) (Huang, 2018) (Dzitkowskik, 2016) (Bobriakov, 2018) (Sheehan, 2017) (Kostadinov, 2018) (Raval, 2018).

Among those, LSTM is a type of Recurrent Neural Network (RNN) which is very well suited for time series forecasting. In general, Artificial Neural Networks (ANN) are used for detecting patterns in the data; in this case, for finding patterns in bitcoin and ether time series with which to predict the next day's bitcoin and ether closing price in dollars.

For a complete explanation on how LSTM work, apart from the above references, see Olah (2015) (this one is very complete and easy to understand), Chen (2017), Bishop (1995), Kostadinov (2017) and Hochreiter and Schmidhuber (1997).

For this section, the full code in Annex 4 is based on Sheehan (2017), who provides a framework with which to train different models and see how they perform, and Coinalysis (2018), mainly, although some tips and ideas have been taken from the rest of the above references. For this task, libraries Tensorflow (2018) and Keras (2018) are used, which include a huge variety of already created classes and objects with which to build deep learning models.

The rest of the authors referenced have been reviewed to assess their methods; among those, Sheehan is the most honest with the data, as he does not incur in tricks such as training the model also in the test dataset, which obviously produces incredible results

because the model is able to see the data and adjust to it many times, instead of training just on the training set and then trying to capture the behavior of the test set. Raval (2018) is an example of this.

LSTM takes some inputs, which are normalized (deep learning models do not perform well with highly variant inputs, for this reason they must be on the same scale). This is one of the concepts developed by Kurzweil (2012): to mimic the mind processes, the AI architecture needs to achieve two things: invariability (that is naturally approached when the system, if scalable, takes a long learning process in very different areas[11]), and redundancy, which can be achieved when enough variables, highly correlated to each other, are presented as inputs to the ANN. Then, it combines the inputs in each of the neurons in the hidden layer, adjusting the weights of these connections optimally, using the Adam Optimizer (see Kingma and Ba (2017)) so the error (in this case the loss function is Mean Squared Error (MSE)) is minimized with respect to the output presented, which in this project is the next day's bitcoin and ether price, separately. An interesting feature of these networks is that they can 'remember' long-term connections among the variables; therefore, allowing for the detection of more complex behaviors in the data. Below, figure 5 summarizes the processes occurring in the network:

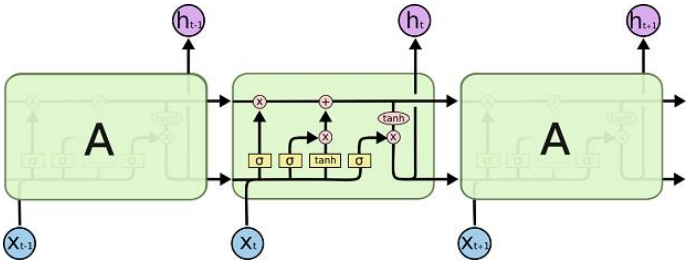


Figure 5. Visualizing the processes of a LSTM Neural Network

Source: Olah (2015)

The upper flow of information (the transversal road on top of each A) is the one that adjusts the connections coming from the lower flow, therefore, it is a transversal pipeline that readjusts itself every iteration of the training period, which enables the network to consider long term dependencies in the data. Note that in this figure the output variable (h) is in the same time as the input variables; this does not hold for the model presented here, in which for an input X_{t-1} the output is h_t (or Y_t, for using traditional machine learning nomenclature).

[11] In this case, the different areas represent different market scenarios, so that the model learns what happens at each concrete combination of the presented variables in the input layer.

The method used in this sub section for forecasting the values for the second period from 2017-10-01 to 2018-03-22 differs significantly from the machine learning procedure in sub section 4.3., as with LSTM the network is trained from 2016-01-01 up to 2017-09-30, and then it uses the connections and weights regarded as optimal for the second period. This means that ARIMA + GARCH (1, 1) has the advantage of being able to change its equation every day also in this second period, while LSTM is not. Nevertheless, this computation requires much more time and computing power, which is not an issue for daily trading but which costs should be considered in case of trying to zoom in the prediction window.

In these types of models, there is a typical problem called overfitting (Brownlee, 2016). This means that the model learns to adjust its weights and neural connections perfectly for fitting the training data, minimizing the error in this step so much that when the model is confronted with test data (second period) it is not able to capture the price movement patterns so well, as it learned too much from the training data, in fact, random noise or fluctuations in the training data are learned as concepts by the model.

This is a weighty problem especially when training and test data differ significantly, which is the case with the task at hand (sharp price movements in the late 2017 and the beginning of 2018 differ significantly from the prices changes in 2016, for example, as seen in the graphs in Annex 4 (1) (2)). To avoid overfitting, a dropout (Srivastava et al., 2014) of 0.25 is selected; then, the number of epochs for training is set to 50, so that the model does not train too much.

For this part of the project, many different parameters have been tested, as well as different types of ANN, like Gradient Recurrent Unit (GRU), which is comparable to LSTM but allows for less complexity. From all models tried, only the best ones for bitcoin and ether are presented; the three of them are LSTM, as this RNN family is the one that most accurately predicts next day's prices.

In the initial model presented by Sheehan (2017), the input variables included are the daily OHLC (Open, High, Low, Close)[12] for both bitcoin and ether, as well as the volumes of both cryptocurrencies. Then, these data are used to predict next day's closing price for ether and bitcoin, separately. Coinalysis (2018) includes more cryptocurrencies for the task, as well as cryptocurrency related news, which enhances the model performance;

[12] Observe that these variables are transformed before being set as inputs for the LSTM model. These transformations are clearly seen in the code by Sheehan (2017); the input variables are therefore Close Price, Volume, Close Off High and Volatility, for each cryptocurrency.

however, as his code is not available, there is no possibility to assess much of the changes he makes with respect to Sheehan (2017) in terms of window selection, which cryptocurrencies to use, and which variables from these cryptos to include in the model (the former used the code of the latter as a basis).

Among the changes (with respect to Sheehan (2017)) tried, cryptocurrencies xmr (Monero) and dash have been included in the model (separately and together); neurons in the hidden layer have been set to 10, 15, 20, 30, 35, 40, 50, 100, however this did not improve much the performance in the test set, showing that increasing the number of neurons too much typically leads to overfitting. A window of 5, 6, 7, 8, 10, 12, 15 and 30 days has been tried, while the best window in terms of minimizing MAPE in the test period was found to be between 5 and 10. Other possibilities explored included adding more hidden layers, trying different activation functions like Leaky ReLU and tanh, and using a diverse range of dropouts.

The decision of which cryptocurrencies to include as inputs for the model was based in the previous portfolio analysis. Dash and Monero have a high correlation with bitcoin and ether, and they are in neighbor clusters (see figures 1 and 2), hence, it makes sense that as they move similarly throughout time, they can be a good predictor of future bitcoin and ether prices.

The first LSTM model, which will be referred to as LSTM (1), has a window of 5 days, includes as inputs the normalized close prices from bitcoin, ether and dash[13], as well as the volatility for each of these three cryptocurrencies; the variable close_off_high (see Annex 4) is included for bitcoin and ether, but not for dash. With respect to Sheehan (2017), the volume of all cryptocurrencies was eliminated from the inputs; however, the number of neurons in the hidden layer was set to 20 as this was found to be an optimal number of units. The dropout was maintained at 0.25; likewise, the activation function used is linear. Figure 6 shows the scheme of the network.

[13] Coinalysis (2018) claims that adding 3 cryptocurrencies' price information the model performs better than with 1 or 2; for this reason, dash was added to LSTM (1), and then xmr was also included for LSTM (2).

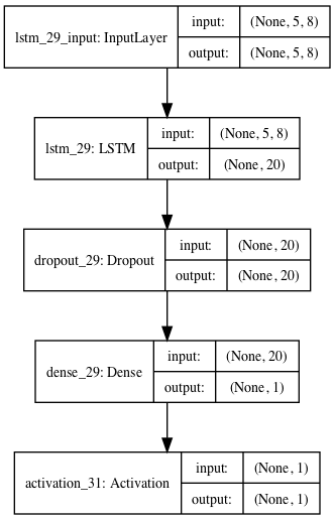


Figure 6. Structure of the LSTM (1) model.

Source: Author.

The second LSTM (LSTM (2)) takes as inputs all the previous information included in LSTM (1), plus XMR's close value (normalized) and volatility. The structure of the model is like the one in figure 6: for both bitcoin and ether, the window is equal to 5 days and there is a single hidden layer with 20 neurons; however, in this case the dropout for ether is 0.35, while for bitcoin it is 0.25.

Table 3 below shows the prediction accuracy, measured as 100 – MAPE, for all LSTM models.

MODEL	Bitcoin	Ether
LSTM (1)	**95.20 %**	**95.25 %**
LSTM (2)	**95.15%**	**95.27%**
LSTM (3)	**95.10%**	**89.63%**

Table 3. Summary of LSTMs' Performance.

Data Sources: Cryptocompare (2018), Coinmarketcap (2018) (1) (2) (3) (4).

LSTM (3) is built with a much simpler architecture, just using as input layer the last prices of bitcoin and ether, respectively. This model is based in Bobriakov (2018), and, despite its lower accuracy, it seems to predict price directions quite well, as seen in figures 8 and 9. The architecture structure for LSTM (3) is the following:

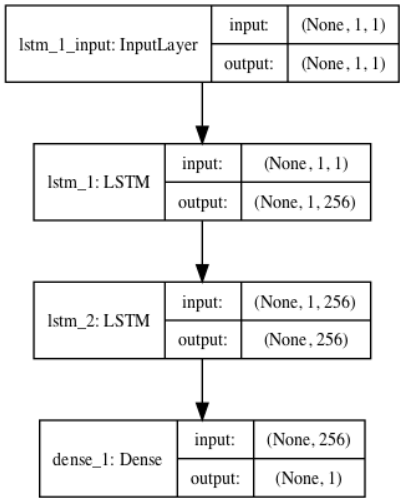


Figure 7. Structure of the LSTM (3) model.

Source: Author.

As it is observable, LSTM (3) does not contain any dropout, and has two hidden layers each containing 256 neurons. This allows for much deeper learning; however, it can lead to overfitting. Furtherly, the trading results of the strategy based in LSTM (3) predictions are assessed, at which point, conclusions are drawn.

There is one important limitation of deep learning with respect to cryptocurrencies' time series. As Kurzweil (2012) explains, the Hierarchical Hidden Markov Models (HHMM), together with the rest of the architecture on which most artificial intelligence algorithms are based, depend on, among other facts, invariant (or slightly variant) data. These models and architecture attempt to imitate how our mind works, based partly on the Pattern Recognition Theory of Mind (Kurzweil, 2012), and it seems that our mind, which is a very accurate predictive machine, is much less successful when facing variant outcomes, due to our naturally linear interpretation of the world.

This explains why the LSTM are not able to capture the price movements from day to day, as these volatile assets change too much. This issue could be probably overtaken if higher frequency data was presented to the networks, as these slight changes would be more easily captured. However, this could cause another objection for real use, as the time required by the model to fit and estimate can be higher than the trading time frequency, and therefore would not be useful. More research should be carried out to arrive at a conclusion, and, in case that paradox persists, find an intelligent way to solve it.

4.5. RESULTS AND TRADING STRATEGY IMPLEMENTATION

For both bitcoin and ether, the ARIMA + GARCH (1, 1) model outperforms LSTM (1), LSTM (2) and LSTM (3), in terms of prediction accuracy. It is interesting to note that other researchers, as McNally (2016) or Amjad and Shah (2016), comparing ARIMA with AI or other ML models, found that the latter forecast better than the former. However, the results of this project show just the opposite. This leads to the conclusion that, by adding: i) GARCH (1, 1) to model prices' volatility, ii) a ML approach that provides the model with more flexibility and the ability to adapt to changes in daily time series' behavior, and iii) a ML approach for selecting the optimal window, the traditional ARIMA model was considerably improved; thus, enhancing the reliability and significance of this project's results.

The next feature of interest from these models is their ability to produce returns for investors, to assess the feasibility and desirability of developing a daily trading strategy based on these algorithms.

The trading strategy for all models is the following. At time t, the prediction for t + 1 for each cryptocurrency is checked; if the predicted price for t + 1 is higher than today's price, a long position is taken. A short position is selected if the opposite to the above happens. In case the predicted price for t + 1 is equal to the cryptocurrency price in t, $\text{position}_t = 0$. Then, the cumulative return of each strategy (one for each model) is calculated as follows: at time t, the cumulative return up to that point is the cumulative return up to t – 1, plus the product of the cryptocurrency's log return between t – 1 and t and the position at time t – 1 (see equation 2). Then, the cumulative return of each of these strategies is compared to the Buy and Hold strategy. For the full code for this sub section, see Annex 5.

Figures 7 and 8 below show the performance of each strategy for bitcoin and ether, respectively:

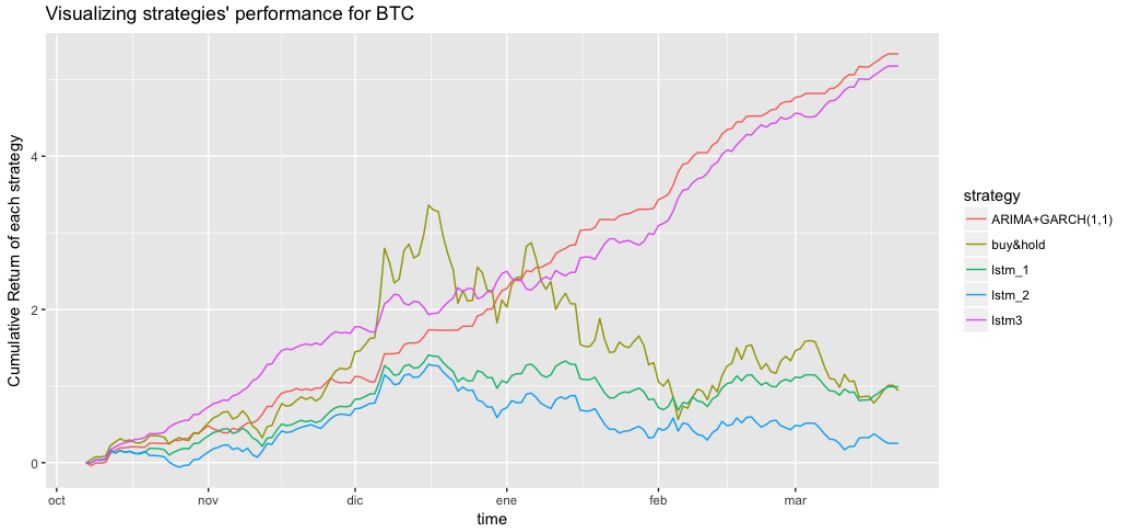


Figure 8. Visualization of trading strategies' performance for bitcoin.

Data Source: Cryptocompare (2018).

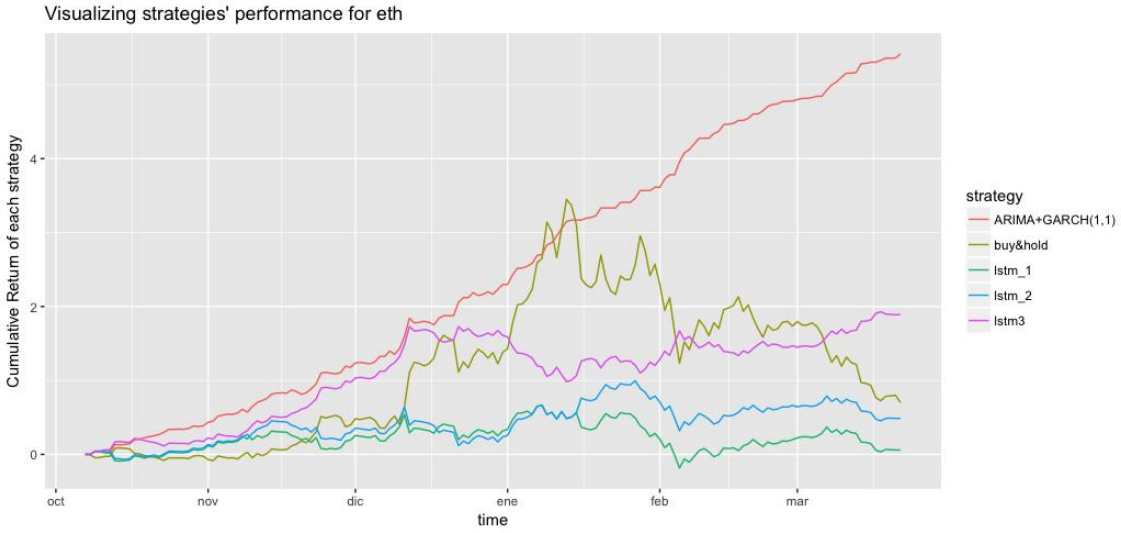


Figure 9. Visualization of trading strategies' performance for ether.

Data Source: Cryptocompare (2018).

From the above graphs, three important conclusions can be drawn. First, the LSTM models do not perform as well as expected, as they are, in the three cases, outperformed by ARIMA + GARCH (1, 1). Second, ARIMA + GARCH (1, 1) does provide a satisfactory cumulative return, which is much higher than the one provided by the Buy & Hold strategy. Furthermore, LSTM (3), which is more akin to the ARIMA + GARCH (1,1) in terms of structure, meaning that its inputs are also only previous prices, outperforms LSTM (1) and (2) in both bitcoin and ether, which shows that feeding the LSTM with more data bias the estimation of next day's price direction.

Table 4 below summarizes the metrics of importance for each of the models under study. For the Profit column (see equation 3), it was assumed that 1,000$ ($Profit_0$) were invested at the beginning of the period; then, profit at time t is calculated as 1 + cumulative return at time t times 1,000$, less 1,000$. The last profit, at day 2018-03-22, is presented. In the below equations, Y_t represents bitcoin or ether price at time t.

$$Cumulative\_Return_t = \sum_{t_0=2}^{t} Position_{t-1} \cdot \ln(\frac{Y_t}{Y_{t-1}})$$

Equation 2. Cumulative Return at time t.

Source: Author.

$$Profit_t = Profit_0 \cdot (1 + Cumulative\_Return_t) - Profit_0$$

Equation 3. Profit at time t.

Source: Author.

Model	Bitcoin			Ether		
	Accuracy	Cum. Ret.	Profit ($)	Accuracy	Cum. Ret.	Profit ($)
ARIMA + GARCH (1, 1)	**98.40%**	**534%**	**5335**	**97.38%**	**542%**	**5416**
LSTM (1)	**95.20%**	**99%**	**992**	**95.25%**	**6%**	**59**
LSTM (2)	**95.15%**	**25%**	**253**	**95.27%**	**49%**	**486**
LSTM (3)	**95.10%**	**518%**	**5176**	**89.63%**	**189%**	**1889**
Buy&Hold	**-**	**94%**	**941**	**-**	**70%**	**702**

Table 4. Summary of Trading Strategies' Performance

Source: Author.

As seen in table 4, the ARIMA + GARCH (1, 1) model generates a cumulative return of 534% and 542% for bitcoin and ether, respectively. While some considerations are missing, as the bid-ask spread, the interest payment when the position is short, and commission fees[14], among others, it can be claimed that the algorithmic trading using this model would have been very successful relative to the Buy & Hold strategy. For the

[14] Commission fees are considered in Annex 5, showing that, taking Binance fees, they are almost negligible and do not affect significantly the cumulative returns of the strategies.

LSTM (1) and (2) models, the same cannot be said, as the only one that outperforms the Buy & Hold strategy is the LSTM (1) for bitcoin.

Notwithstanding this, LSTM (3) beats the Buy & Hold strategy for both bitcoin and ether, with especially good results in the case of the former. Though, a consideration has still to be made, as Bobriakov's (2018) code makes a callback in which the loss function is optimized first in the training period and then checked in the validation (test, in this case) period, contrasting with the approach followed in LSTM (1) and (2). Therefore, LSTM (3)'s results are not as encouraging as ARIMA + GARCH (1,1)'s ones.

As mentioned before, the ARIMA + GARCH (1, 1) model has an advantage over LSTM models, as it readjusts itself every iteration, while LSTMs learn during the training period and then predict the full 167 days' prices (test period), except for LSTM (3). This means that they cannot constantly learn with new information in the test period, as opposed to the ARIMA + GARCH (1, 1).

This leads to a reflection, which is linked to the ideas developed by Taleb (2004) (2007) (2012). Anything included in a mathematical model, will only account for what has already happened, past information is the only one available for every model. This past information will always be insufficient, due to the time dimension, no matter how much time of events is included in the model, as there are virtually infinite events still to occur. For this reason, time series forecasting is such a hard exercise; in the case of LSTM, we can teach the model to almost perfectly understand the connections between the variables with respect to the output variable of interest from past data, however, this backward explanation (Taleb, 2007) does not necessarily hold in the future, and this is what makes most markets weak form efficient in the long run (Degutis and Novickyte, 2014) (Investopedia, 2018).

This is especially important for the LSTM, as it stops learning before the test period, therefore it cannot incorporate new information to the model constantly. On the other hand, ARIMA + GARCH (1, 1) can capture new behaviors of the time series, as it can adapt itself every iteration to new data: it does never stop learning; for that reason, the time dimension does not have such a strong effect over ARIMA + GARCH (1,1)'s forecasts. However, it is noticeable that the performance of such a model is only satisfactory when dealing with daily forecasts, even with constant learning it is very difficult to assess how the market will move in the next 5-10 days (Sheehan, 2017).

4.5.1. Limitations, Future Work and Improvements

Although the performance of both ARIMA + GARCH (1, 1) and LSTM are, to a different extent, not disappointing, much work can be done to improve the models' performance and therefore achieve a more successful trading strategy for cryptocurrencies. First, using daily data is not necessarily a good option, as cryptocurrency prices move much in a day, and therefore differences from day to day may be too big. Moreover, in terms of designing a trading strategy, the time lag between 00.00h and the time at which the orders are executed, can limit the potential return achievable. Though, a daily trading strategy has the advantage of having enough time to make next day's prices predictions.

For achieving a successful strategy, it is probable that using 10, 20 or 30 seconds' data, or even minute data, provides better results, as the models have much more information (which in the short term is less variant) with which to learn, and these models could be more easily coordinated with a program that performs high frequency trading. A proof of this is given by Huang (2018), who uses data on a 5 minutes' basis: "the input spans across 1280 minutes and the output covers 80 minutes". For that, he uses the Poloniex API, which can be also used for trading once the models have forecasted. A more powerful computer would be needed for this.

Another important limitation of this project is the omission of the variable "investor sentiment", which, given the bibliography reviewed (see sub section 4.1.), is a very important predictor of crypto prices. As a matter of fact, Colianni, Rosales and Signorotti (2015) achieved a classification accuracy of up to 95%, Stenqvist and Lönnö (2017) 79% and Kim et al. (2016) 79.57%, all of them using, in different ways, investor sentiment data. This would be beyond the scope of this work, although some exploration on the possibilities for retrieving these data has been carried out, to evaluate the feasibility of including that into the model.

Given the limitations of the Twitter API, which does not allow public users to retrieve tweets older than a week, the only way to approach this is by developing higher frequency models; the idea would be to retrieve tweets on a 5-minute basis, and then, using some sentiment classification algorithm like the ones in NLTK (Natural Language Toolkit, 2017) package, to create numerical variables expressing how positive, neutral and negative the market is about a cryptocurrency; these variables would be then used for forecasting, together with the past prices.

A similar approach to this is the one by Coinalysis (2018), although he uses news headlines[15] together with past price information for his 'LSTM Extended Model', with great success, as it improves significantly the predictive ability of the model. Given the huge research regarding sentiment analysis and cryptocurrency prices, as well as the predictive power of this variable, it can add much to the model, enabling it to have more of the information it needs to make accurate predictions. In the case of ARIMA + GARCH (1,1) this would mean to eliminate the variable investor sentiment from ε, therefore getting closer to the desired model with an error only representing noise.

Moreover, this type of asset is, as described at the beginning of the writing, very speculative, and highly dependent on news and rumors, which ultimately lead to sharp movements in prices. With the intuition in mind that there may be a time lag between people's reaction to the news and rumors (sentiment in the market) and price changes, there may be an arbitrage possibility by using deep learning models which include these variables: sentiment taken from Twitter, Reddit, or other media (the more data the better) and news headlines taken as Coinalysis (2018), as well as from important newspapers' APIs. This would probably improve the prediction accuracy of the models and therefore increase their reliability for implementing a trading strategy.

Other variable that could be included in both ARIMA + GARCH (1, 1) and LSTM are the interest in bitcoin and ether, using Google searches information; however, Google Trends provides daily data for only 90 days, therefore other methods for mining that data should be used. The relationship between weekly Google searches (from now on, 'public interest') and weekly bitcoin and ether prices was examined, but not presented as it was not useful for daily forecasting; however, there is a clear influence between these variables, and therefore attempts to get daily data in that direction should be carried out.

Additionally, the portfolio management strategy developed in section 3.2. can be significantly improved by designing an AI system that learns to automatically readjust the portfolio weights with new information on developer and community support, investor sentiment towards each cryptocurrency, and the rest of the variables used for allocating the portfolio weights in 3.2. This can significantly increase the trading strategy returns, since the portfolio can adapt itself to the new market conditions.

[15] Available at <www.newsnow.co.uk> [Accessed March 25th, 2018]

5. CONCLUSION

The results obtained in this project are encouraging in various aspects, as they contribute to the development of fruitful trading strategies for cryptocurrencies. First, regarding the portfolio selection methodology, the data analyzed clearly supports the cryptocurrencies' classification by Ryan (2017) and tuned by me, based on my own experience and knowledge. The hierarchical clustering method used, based in ACF, splits cryptocurrencies in roughly the same groups as expected. Nevertheless, for a more successful portfolio management strategy, the next step is to improve it by designing an AI system that automatically adjusts the portfolio weights daily, thus maximizing investor's returns.

Regarding the predictive models, ARIMA + GARCH (1,1) provides the best results, both in terms of accuracy (98.40% for bitcoin and 97.38% for ether) and cumulative return, therefore, if one model had to be chosen for implementing the trading strategy described in this project, it would be this one. This shows that all the contributions to this model performed in this project (see page 30) proved successful, as these results contrast with other researchers' ones. The investor would have achieved, from 2017-10-07 to 2018-03-22 a return of 534% and 542% in bitcoin and ether, respectively, both of which are much higher than the Buy & Hold strategy (94% and 70%).

Among the LSTMs, the only one that clearly outperforms the Buy & Hold strategy for both bitcoin and ether is LSTM (3), thus showing that feeding the RNN with more inputs (as in LSTM (1) and (2)) improves forecast accuracy, but reduces the cumulative return, which means that price directions are better assessed by LSTM (3) than both LSTM (1) and (2). Nonetheless, LSTM (3) was calculated by optimizing the network's connections to minimize its loss function both in the training and the test period; ergo, its results are less reliable than the ones of LSTM (1), (2) and ARIMA + GARCH (1,1).

The main limitations of this trading strategy deal with not considering the interest costs when going short, the time span between prediction and order execution and the bid/ask spread, among others. Likewise, the main limitations for all the models presented are the lack of representation of the variables investors' sentiment, public interest and cryptocurrency-related news. These three variables will be included in future work, to develop the presented models and the trading strategy further.

For that, given the APIs limitations for retrieving these data, models should be adapted to higher frequency data. This could have the additional advantage of reducing the impact

of time dimension, as less invariant and more redundant data would be presented to the ML and AI algorithms. For implementing this properly on a real-time automatic algorithmic trading strategy, a more powerful computer would be needed.

After reviewing all the idiosyncratic reasons for investing on Blockchain technology, and given the desirability of its development for the progress of institutions and societies, this project sheds some light also on the other motives for doing so, that is, the economic ones. The cryptocurrency market's characteristics show that it is still an immature market, with much potential growth still to come, which will largely depend on regulation and on the success and development of current Blockchain implementations. Though, by providing faster, cheaper, safer, more inclusive, transparent and scalable networks systems, this technology has the potential to add much value compared to current practices, which is expected to be captured by these Blockchains' tokens.

This project shows how investing on and financing Blockchain technology, which has the potential to improve several of society aspects, and is therefore generally considered beneficial given the literature reviewed, can be in fact a profitable investment opportunity, as ML and AI models for beating the market were developed, thus showing that the cryptocurrency market is not weak-form efficient in the short-run.

BIBLIOGRAPHY

- AlphaMacro (2018) Dynamic WARIMAX-gjrGARCH Market Strategy. *AlphaMacro* [blog] January 27th. Available at <https://www.alphamacrocm.com/all/warimax-garch-strategy> [Accessed Feb. 1st, 2018]
- Amjad, M. J. and Shah, D. (2016) Trading Bitcoin and Online Time Series Prediction. *Proceedings of the Time Series Workshop at NIPS 2016, PMLR* 55,1-15.
- Antonopoulos, A. M. (2017) *Mastering Bitcoin: Programming the Open Blockchain.* 2nd Edition. USA: O'Reilly Media.
- Aungiers, J. (2017) Multidimensional LSTM Networks to Predict Bitcoin Price. *Jakob-Aungiers* [blog] July 15th. Available at <http://www.jakob-aungiers.com/articles/a/Multidimensional-LSTM-Networks-to-Predict-Bitcoin-Price> [Accessed March 20th, 2018]
- *Banking on Bitcoin* (2016) [Documentary film] directed by Cannucciari, C. USA: Periscope Entertainment.
- Barro, R. (1979) Money and the Price Level under the Gold Standard. *Economic Journal* 89 (353), 13-33.
- Berlinger, E., Illés, F., Badics, M., Banai, Á., Daróczi, G., Dömötör, B., Gabler, G., Havran, D. and Juhász, P. (2015) *Mastering R for Quantitative Finance: Use R to optimize your trading strategy and build up your own risk management system*. Birmingham, UK: Packt Publishing.
- Binsumi (2016) Neural Networks and Bitcoin. *Medium* [blog] July 1st. Available at <https://medium.com/@binsumi/neural-networks-and-bitcoin-d452bfd7757e> [Accessed March 27th, 2018]
- Bishop, C. M. (1995) *Neural Networks for Pattern Recognition*. UK: Clarendon Press-Oxford.
- Bobriakov, I. (2018) Bitcoin price forecasting with deep learning algorithms. *Medium* [blog] March 5th. Available at <https://medium.com/activewizards-machine-learning-company/bitcoin-price-forecasting-with-deep-learning-algorithms-eb578a2387a3> [Accessed March 20th, 2018]

- Bocconi Students Investment Club (2017). A Markowitz Walk Down Crypto-land: Modern Assets for Modern Portfolios. *Bocconi Students Investment Club* [online]. Available at: <http://www.bsic.it/markowitz-walk-crypto-land-modern-assets-modern-portfolios/> [Accessed Feb. 15th, 2018]
- Brownlee, J. (2016) Overfitting and Underfitting with Machine Learning Algorithms. *Machine Learning Mastery* [online] Available at <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/> [Accessed March 28th, 2018]
- Brynjolfsson, E. & McAffee, A. (2014) *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. USA: Norton.
- Cadwalladr, C. and Graham-Harrison, E. (2018) 'Revealed: 50 Million Facebook profiles harvested for Cambridge Analytica in major data breach', *The Guardian*, March 17th [online newspaper] Available at <https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election> [Accessed April 10th, 2018]
- Chen, E. (2017) Exploring LSTMs. *Blog E. Chen* [blog] May 30th. Available at <http://blog.echen.me/2017/05/30/exploring-lstms/> [Accessed March 28th, 2018]
- Chen, M., Narwal, N. and Schultz, M. (2017) Predicting Price Changes in Ethereum. *Stanford University* [online] Available at <http://cs229.stanford.edu/proj2017/final-reports/5244039.pdf> [Accessed March 06th, 2018]
- Chu, J., Nadarajah, S. and Chan, S. (2015) Statistical Analysis of the Exchange Rate of Bitcoin. *PLoS ONE* 10 (7): e0133678.
- Ciaian, P., Rajcaniova, M. and Kancs, d'A. (2015) The Economics of Bitcoin Price Formation. *Applied Economics* 48 (19), 1799-1815.
- Coinalysis (2018) Predict cryptocurrency prices based on news and historical price data. *Wordpress* [blog] Jan. 22nd. Available at <https://coinalysis.wordpress.com/2018/01/22/predict-cryptocurrency-prices-based-on-news-and-historical-price-data-2/> [Accessed March 25th, 2018]
- Coingecko (2018) *Coingecko* [online] Available at <https://www.coingecko.com/en> [Accessed Feb. 18th, 2018]

- Coinmarketcap (2018) (1) Historical Data for Bitcoin. *Coinmarketcap* [online] Available at <https://coinmarketcap.com/currencies/bitcoin/historical-data/?start=20130428&end=> [Accessed March 28th, 2018]
- Coinmarketcap (2018) (2) Historical Data for Ethereum. *Coinmarketcap* [online] Available at <https://coinmarketcap.com/currencies/ethereum/historical-data/?start=20130428&end=> [Accessed March 28th, 2018]
- Coinmarketcap (2018) (3) Historical Data for Dash. *Coinmarketcap* [online] Available at <https://coinmarketcap.com/currencies/dash/historical-data/?start=20130428&end=> [Accessed March 28th, 2018]
- Coinmarketcap (2018) (4) Historical Data for Monero. *Coinmarketcap* [online] Available at <https://coinmarketcap.com/currencies/monero/historical-data/?start=20130428&end=> [Accessed March 28th, 2018]
- Coinmarketcap (2018) Percentage of Total Market Capitalization (Dominance). *Coinmarketcap* [online] Available at <https://coinmarketcap.com/charts/> [Accessed Feb. 20th, 2018]
- Colianni, S., Rosales, S. and Signorotti, M. (2015) Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis. *Stanford University* [online] Available at <http://cs229.stanford.edu/proj2015/029_report.pdf> [Accessed Dec. 19th, 2017]
- Collier, A. B. (2017) Clustering Time Series Data. *Exegetic* [blog] April 25th. Available at <http://www.exegetic.biz/blog/2017/04/clustering-time-series-data/> [Accessed Feb. 15th, 2018]
- Cryptocompare (2018) Coins List. *Cryptocompare* [online] Available at <https://www.cryptocompare.com/coins/list/USD/1> [Accessed Feb. 16th, 2018]
- Degutis, A. and Novickyte, L. (2014) The Efficient Market Hypothesis: A Critical Review of Literature and Methodology. *Economika* 93 (2), 7-23.
- Dzitkowskik (2016) StockPredictionRnn. *Github* [Repository] May 21st. Available at <https://github.com/dzitkowskik/StockPredictionRNN> [Accessed March 15th, 2018]
- Fantazzini, D., Nigmatullin, E., Sukhanovskaya, V. and Ivliev, S. (2016) Everything you always wanted to know about bitcoin modelling but were afraid to ask. *Munich Personal RePEc Archive (MPRA)*, Paper No. 71946

[online] Available at <https://mpra.ub.uni-muenchen.de/71946/1/MPRA_paper_71946.pdf> [Accessed Feb. 02nd, 2018]

- Georgoula, I., Pournarakis, D., Bilanakos, C., Sotiropoulos, D. N. and Giaglis, G. M. (2015) Using Time - Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices. *MCIS 2015 Proceedings*, 20. [online] Available at <https://pdfs.semanticscholar.org/05f3/7fcb95488402bb9e4d0d5a291e1338de41a6.pdf> [Accessed Feb. 15th, 2018]
- Ghalanos, A. (2018) Package 'rugarch'. *CranR-Project* [online] Available at <https://cran.r-project.org/web/packages/rugarch/rugarch.pdf> [Accessed March 28th, 2018]
- Google Trends (2018) Google Trends: Explore. *Google Trends* [online] Available at <https://trends.google.es/trends/explore> [Accessed Feb. 19th, 2018]
- Graham, B. (2007) *El inversor inteligente.* 5a Edición. Barcelona, Spain: Deusto.
- Greaves, A. and Au, B. (2015) Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin. *Stanford University* [online] Available at <http://snap.stanford.edu/class/cs224w-2015/projects_2015/Using_the_Bitcoin_Transaction_Graph_to_Predict_the_Price_of_Bitcoin.pdf> [Accessed March 03rd, 2018]
- Grinold, R. C. & Kahn, R. N. (1999) *Active Portfolio Management: A Quantitative Approach for Providing Superior Return and Controlling Risk*. New York, USA: McGraw-Hill.
- Hegazy, K. and Mumford, S. (2016) Comparative Automated Bitcoin Trading Strategies. *Stanford University* [online] Available at <http://cs229.stanford.edu/proj2016/report/MumfordHegazy-ComparitiveAutomatedBitcoinTradingStrategies-report.pdf> [Accessed March 03rd, 2018]
- Heinz, S. (2017) A simple deep learning model for stock price prediction using TensorFlow. *Medium* [blog] Nov. 9th. Available at <https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877> [Accessed March 26th, 2018]
- Hencic, A. and Gouriéroux, C. (2015) Noncausal Autoregressive Modeling Application to Bitcoin/USD Exchange Rates. *Econometrics of Risk, Studies in Computational Intelligence* 583, 17-39.

- Hochreiter, S. & Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation* 9 (8), 1735-1780
- Huang, S. (2018) Predicting Cryptocurrency Price with Tensorflow and Keras. *Medium* [blog] Jan. 1st. Available at <https://medium.com/@huangkh19951228/predicting-cryptocurrency-price-with-tensorflow-and-keras-e1674b0dc58a> [Accessed March 22nd, 2018]
- Hyndman, R. J. (2017) Package Forecast. C*ranR-Project* [online] Available at <https://cran.r-project.org/web/packages/forecast/forecast.pdf> [Accessed Feb. 27th, 2018]
- Investopedia (2018) Weak, Semi-Strong and Strong EMH. *Investopedia* [online] Available at <https://www.investopedia.com/exam-guide/cfa-level-1/securities-markets/weak-semistrong-strong-emh-efficient-market-hypothesis.asp> [Accessed April 01st, 2018]
- Kaggle (2018) Bitcoin Historical Data. *Kaggle* [online] Available at <https://www.kaggle.com/c/bitcoin-price/data> [Accessed 22 Jan. 2018]
- Kahneman, D. (2012) *Pensar Rápido, Pensar Despacio*. Barcelona, Spain: Penguin Random House Grupo Editorial.
- Keras (2018) Keras Documentation. *Keras* [online] Available at <https://keras.io/> [Accessed Feb. 15th, 2018]
- Khandelwal, I., Adhikari, R. and Verma, G. (2015) Time Series Forecasting using Hybrid ARIMA and ANN Models based on DWT Decomposition. *Elsevier: Procedia Computer Science* 48 (2015), 173-179.
- Kim, Y. B., Kim, J. G., Im, J. H., Kim, T. H., Kang, S. J., Kim, C. H. (2016) Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. *PLoS ONE* 11 (8): e0161197.
- Kingma, D. P. and Ba, J. (2017) Adam: A Method for Stochastic Optimization (last revision, version 9). *Cornell University Library* [online] Available at <https://arxiv.org/abs/1412.6980> [Accessed March 25th, 2018]
- Kostadinov, S. (2017) How Recurrent Neural Networks Work. *Medium* [blog] Dec. 2nd. Available at <https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7> [Accessed March 27th, 2018]

- Kostadinov, S. (2018) Bitcoin price prediction using LSTM. *Medium* [blog] Feb. 1st. Available at <https://towardsdatascience.com/bitcoin-price-prediction-using-lstm-9eb0938c22bd> [Accessed March 27th, 2018]
- Kristofuek, L. (2015) What Are the Main Drivers of the Bitcoin Price? Evidence from Wavelet Coherence Analysis. *PLoS ONE* 10 (4): e0123923.
- Kuo Chuen, D. L. (2015) *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data.* London: Elsevier.
- Kurzweil, R. (2012) *How to Create a Mind: The Secret of Human Thought Revealed*. USA: Viking Penguin.
- MacDonell, A. (2014) Popping the Bitcoin Bubble: An application of log-periodic power law modeling to digital currency. *University of Notre Dame* [online] Available at <https://economics.nd.edu/assets/134206/mac_donell_popping_the_biticoin_bubble_an_application_of_log_periodic_power_law_modeling_to_digital_currency.pdf> [Accessed Feb. 26th, 2018]
- Madan, I., Saluja, S. and Zhao, A. (2014) Automate Bitcoin Trading via Machine Learning Algorithms. *Stanford University* [online] Available at <http://cs229.stanford.edu/proj2014/Isaac%20Madan,%20Shaurya%20Saluja,%20Aojia%20Zhao,Automated%20Bitcoin%20Trading%20via%20Machine%20Learning%20Algorithms.pdf> [Accessed March 03rd, 2018]
- Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, 7 (1), 77-91.
- McFarlane, G. (2018) The barbell investment strategy. *Investopedia* [online] Available at <https://www.investopedia.com/articles/investing/013114/barbell-investment-strategy.asp> [Accessed Feb. 16th, 2018]
- McNally, S. (2016) Predicting the price of Bitcoin using Machine Learning. *National College of Ireland* [online] Available at <http://trap.ncirl.ie/2496/1/seanmcnally.pdf> [Accessed Feb. 22nd, 2018]
- Montero, P. and Vilar, J. A. (2014) TSclust: An R Package for Time Series Clustering. *Journal of Statistical Software*, 62 (1).
- Nakamoto, S. (2008) Bitcoin: A peer to peer Electronic Cash System. *Bitcoin Org*. [online] Available at <https://bitcoin.org/bitcoin.pdf> [Accessed Feb. 7th, 2018]

- Natural Language Toolkit (2017) Natural Language Toolkit Documentation. *Natural Language Toolkit* [online] <https://www.nltk.org/> [Accessed April 10th, 2018]
- Olah, C. (2015) Understanding LSTM Networks. *Github* [blog] August 27th. Available at <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> [Accessed March 20th, 2018]
- PiSimo (2017) BitcoinForecast. *Github* [Repository] Feb. 3rd. Available at <https://github.com/PiSimo/BitcoinForecast> [Accessed March 26th, 2018]
- Pollock, D. (2018) 'Why is Bitcoin in Such a Bad Way?', *Cointelegraph*, Feb. 02nd [online newspaper] Available at <https://cointelegraph.com/news/black-january-why-is-bitcoin-in-such-a-bad-way> [Accessed Feb. 16th, 2018]
- Poyser, O. (2017) Exploring the determinants of Bitcoin's price: an application of Bayesian Structural Time Series. *Cornell University Library* [online] Available at <https://arxiv.org/pdf/1706.01437.pdf> [Accessed Jan. 15th, 2018]
- Preukschat, A., Kuchkovsky, C., Gómez Lardies, G., Díez García, D. & Molero, I. (2017) *Blockchain: La revolución industrial de internet*. 4th ed. Barcelona, Spain: Gestión 2000 - Grupo Planeta.
- Raval, S. (2018) A Deep Learning Approach to Predicting Cryptocurrency Prices. *Github* [Repository] Jan. 1st. Available at <https://github.com/llSourcell/ethereum_future/blob/master/A%20Deep%20Learning%20Approach%20to%20Predicting%20Cryptocurrency%20Prices.ipynb> [Accessed April 03rd, 2018]
- Ryan, J. (2017) Building a Cryptocurrency Portfolio. *Medium* [blog] October 12th. Available at <https://medium.com/wealthrituals/building-a-cryptocurrency-portfolio-43c07863513d> [Accessed Feb. 12th, 2018]
- Shah, D. and Zhang, K. (2014) Bayesian regression and Bitcoin. *Cornell University Library* [online] Available at < https://arxiv.org/pdf/1410.1231.pdf> [Accessed Dec. 18th, 2017]
- Sharma, R. (2017) Can Government Regulation Affect Bitcoin Prices? *Investopedia* [online] Available at <https://www.investopedia.com/news/can-government-regulation-affect-bitcoin-prices/> [Accessed Feb. 16th, 2018]
- Sheehan, D. (2017) Predicting Cryptocurrency Prices with Deep Learning. *Github* [blog] Nov. 20th. Available at

<https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-prices-with-deep-learning/> [Accessed Jan. 29th, 2018]

- Shiller, R. J., and Akerlof, G. A. (2010) *Animal Spirits: How Human Psychology Drives the Economy, and Why It Matters for Global Capitalism*. United Kingdom: Princeton University Press.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929-1958.
- Stenqvist, E. and Lönnö, J. (2017) *Predicting Bitcoin price fluctuation with Twitter sentiment analysis*. Non-published Master thesis. Stockholm: KTH Royal Institute of Technology: School of Computer Science and Communication.
- Taleb, N. N. (2004) *Fooled by Randomness. The Hidden Role of Chance in Life and in the Markets*. USA: Random House.
- Taleb, N. N. (2007) *The Black Swan*. New York, USA: Random House.
- Taleb, N. N. (2012) *Antifragile: Things That Gain from Disorder*. New York, USA: Random House.
- Tapscott, D. & Tapscott, A. (2016) *Blockchain Revolution: how the technology behind bitcoin is changing money, business, and the world*. New York, USA: Portfolio/Penguin.
- Tarnopolski, M. (2017) Modelling the price of Bitcoin with geometric fractional Brownian motion: A Monte Carlo Approach. *Cornell University Library* [online] Available at < https://arxiv.org/abs/1707.03746> [Accessed Dec. 20th, 2017]
- Tensorflow (2018) *Tensorflow* [online] Available at <https://www.tensorflow.org/> [Accessed March 15th, 2018]
- Thaler, R. H., & Sunstein, C, R. (2008) *Nudge: Improving decisions about wealth, health and happiness*. Connecticut, US: Yale University Press.
- Tsay, R. (2013) *An Introduction to Analysis of Financial Data with R*. New Jersey, USA: John Wiley & Sons, Inc.
- Tversky, A. and Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. *Science, New Series*, 185 (4157), 1124-1131.
- Tversky, A. and Kahneman, D. (1979). Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47 (2), 263-292.

- Voronin, S. and Partanen, J. (2013) Price Forecasting in the Day-Ahead Energy Market by an Iterative Method with Separate Normal Price and Price Spike Frameworks. *Energies* 2013 (6), 5897-5920.
- Williams-Grut, O. (2018) 'Here are all the theories explaining the crypto market crash', *Business Insider UK*, Jan. 17th [online newspaper] Available at <http://uk.businessinsider.com/bitcoin-cryptocurrency-market-crash-explained-causes-2018-1> [Accessed Feb. 16th, 2018]
- Yermack, D. (2013) Is Bitcoin a Real Currency? *National Bureau of Economic Research*: Working Paper 19747.
- Zhu, Y., Dickinson, D. and Li, J. (2017) Analysis on the influence factors of Bitcoin's price based on VEC model. *Springer: Financial Innovation* 3: 3 [online] Available at <https://link.springer.com/content/pdf/10.1186%2Fs40854-017-0054-0.pdf> [Accessed Feb. 14th, 2018]

FIGURES INDEX

TABLES INDEX

EQUATIONS INDEX

ACRONYMS INDEX

ACF: Autocorrelation Function

AI: Artificial Intelligence

ANN: Artificial Neural Network

API: Application Programming Interface

ARIMA: Autoregressive Integrated Moving Average

ARIMAX: ARIMA with External Regressors

BTC: bitcoin

CBOE: Chicago Board Options Exchange

CME: Chicago Mercantile Exchange

EC: Empirical Conditional Distribution

ETH: ether

GARCH: Generalized Autoregressive Conditional Heteroskedasticity

GLM: Generalized Linear Model

GRU: Gradient Recurrent Unit

HHMM: Hierarchical Hidden Markov Models

ICO: Initial Coin Offering

LDA: Linear Discriminant Analysis

LR: Logistic Regression

LSTM: Long Short-Term Memory

MACD: Moving Average Convergence Divergence

MAID or maid: Maidsafe

MAPE: Mean Absolute Percentage Error

ML: Machine Learning

MSE: Mean Squared Error

NLTK: Natural Language Toolkit

OHLC: Open, High, Low, Close

P2P: Peer-to-Peer

RF: Random Forest

RNN: Recurrent Neural Network

RRL: Recurrent Reinforcement Learning

RSI: Relative Strength Index

SVM: Support Vector Machine

XMR: Monero

XRP: Ripple

ZEC: Zcash

ANNEX 1

```r
#portfolio
#install.packages('readr')
library(readr)
temp = list.files(pattern="*.csv")
myfiles = lapply(temp, read.csv)
names = c("")

# list all csv files from the current directory
list.files(pattern=".csv$") # use the pattern argument to define a com
mon pattern  for import files with regex. Here: .csv
```

```
##  [1] "bat.csv"  "btc.csv"  "dash.csv" "eos.csv"  "eth.csv"  "iot.cs
v"
##  [7] "lsk.csv"  "ltc.csv"  "maid.csv" "neo.csv"  "xem.csv"  "xmr.cs
v"
## [13] "xrp.csv"  "zec.csv"
```

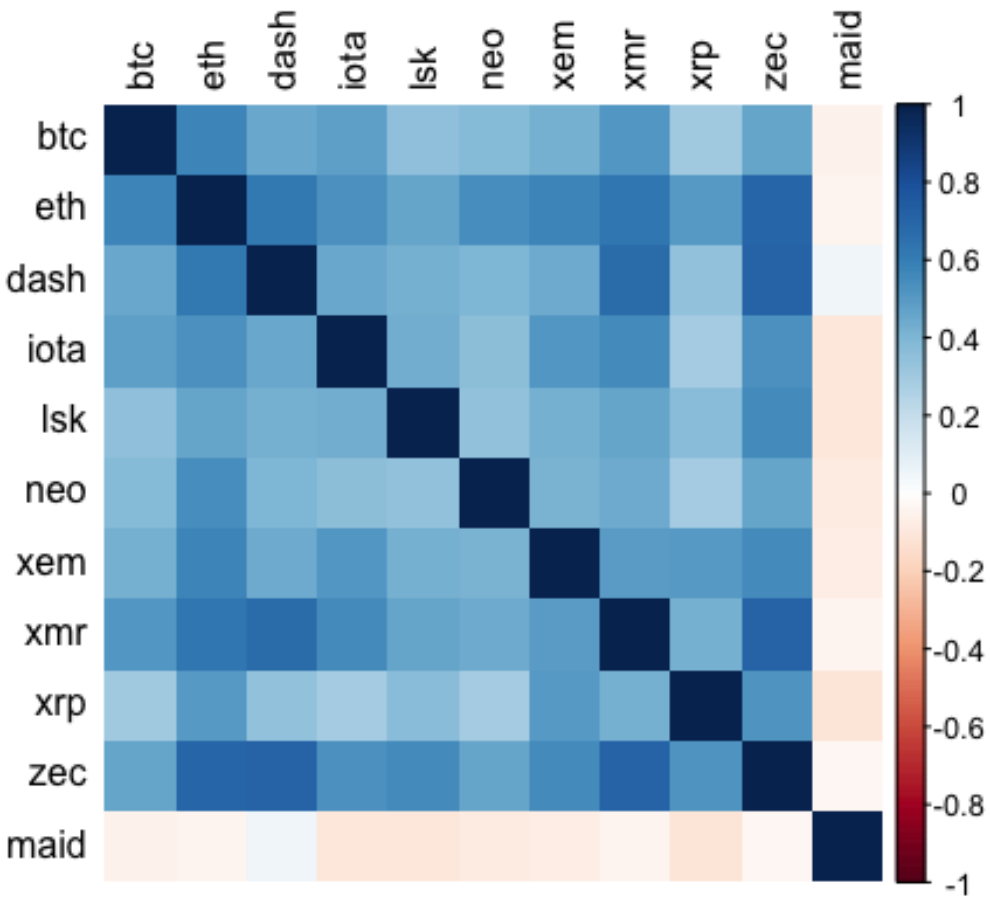
```r
# create a list from these files
list.filenames<-list.files(pattern=".csv$")
list.filenames
```

```
##  [1] "bat.csv"  "btc.csv"  "dash.csv" "eos.csv"  "eth.csv"  "iot.cs
v"
##  [7] "lsk.csv"  "ltc.csv"  "maid.csv" "neo.csv"  "xem.csv"  "xmr.cs
v"
## [13] "xrp.csv"  "zec.csv"
```

```r
# create an empty list that will serve as a container to receive the i
ncoming files
list.data<-list()

# create a loop to read in your data
for (i in 1:length(list.filenames))
{
  list.data[[i]]<-read.csv(list.filenames[i])
}

# add the names of your data to the list
names(list.data)<-list.filenames
da <- list.data
bat = data.frame(da[1])
btc = data.frame(da[2])
dash = data.frame(da[3])
#eos = data.frame(da[4])#we do not include it
eth = data.frame(da[5])
iota = data.frame(da[6])
lsk = data.frame(da[7])
ltc = data.frame(da[8])
maid = data.frame(da[9])
neo = data.frame(da[10])
xem = data.frame(da[11])
xmr = data.frame(da[12])
```

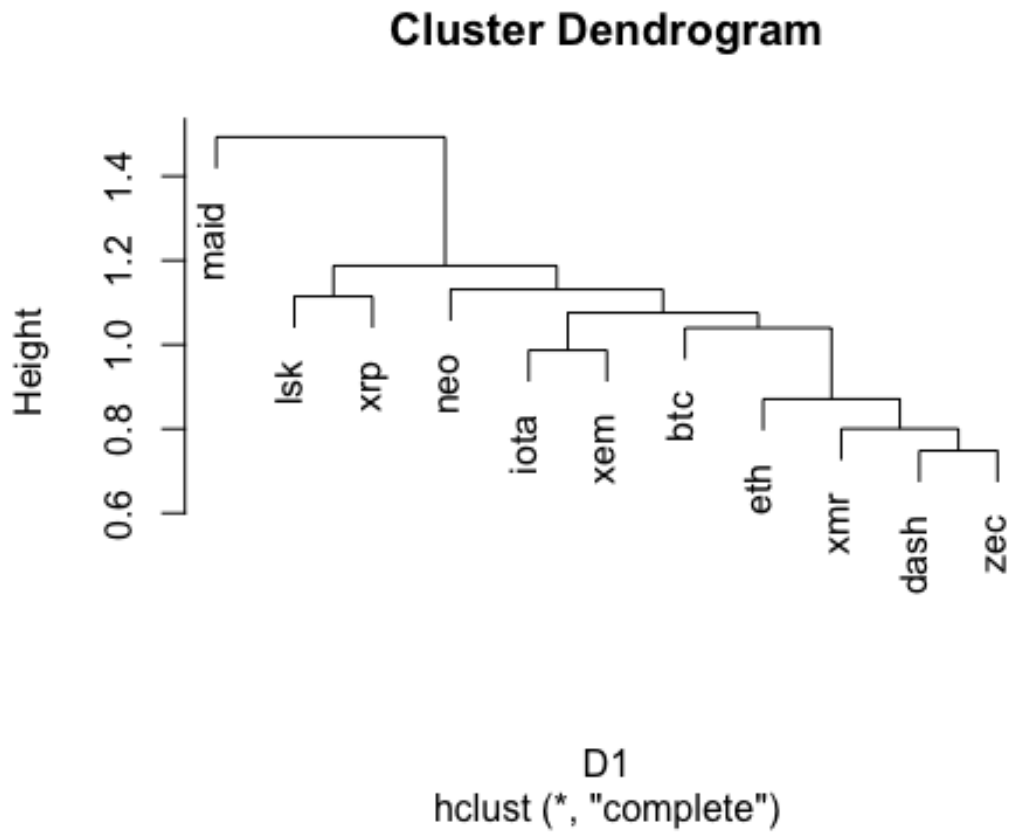
```
xrp = data.frame(da[13])
zec = data.frame(da[14])


#close_prices = data.frame(lapply(da, '[[', 3))

bat = bat[ , 2:5]
btc = btc[ , 2:5]
dash = dash[ , 2:5]
#eos = eos[ , 1:2]
eth = eth[ , 2:5]
iota = iota[ , 2:5]
lsk = lsk[ , 2:5]
ltc = ltc[ , 2:5]
neo = neo[ , 2:5]
xem = xem[ , 2:5]
xmr = xmr[ , 2:5]
xrp = xrp[ , 2:5]
zec = zec[ , 2:5]
maid = maid[ , 2:5]


bat$volatility = bat[,3] - bat[,4]
btc$volatility = btc[,3] - btc[,4]
dash$volatility = dash[,3] - dash[,4]
#we cannot use eos because there is no data about its volatility (high
-low)
eth$volatility = eth[,3] - eth[,4]
iota$volatility = iota[,3] - iota[,4]
lsk$volatility = lsk[,3] - lsk[,4]
ltc$volatility = ltc[,3] - ltc[,4]
neo$volatility = neo[,3] - neo[,4]
xem$volatility = xem[,3] - xem[,4]
xmr$volatility = xmr[,3] - xmr[,4]
xrp$volatility = xrp[,3] - xrp[,4]
zec$volatility = zec[,3] - zec[,4]
maid$volatility = maid[,3] - maid[,4]

#for different reasons, we will not use nor eos nor bat; the first one
because it is still on ICO and most importantly there is no data for U
SD High and USD Low daily.
#for BAT we do not have sufficient data...

#the one with less data is neo, with 197 observations. Therefore, we s
hould take the last 197 observations of all coins
n = 196

btc = btc[(nrow(btc)-n):nrow(btc),]
dash = dash[(nrow(dash)-n):nrow(dash),]
eth = eth[(nrow(eth)-n):nrow(eth),]
iota = iota[(nrow(iota)-n):nrow(iota),]
lsk = lsk[(nrow(lsk)-n):nrow(lsk),]
ltc = ltc[(nrow(ltc)-n):nrow(ltc),]
xem = xem[(nrow(xem)-n):nrow(xem),]
```

```r
xmr = xmr[(nrow(xmr)-n):nrow(xmr),]
xrp = xrp[(nrow(xrp)-n):nrow(xrp),]
zec = zec[(nrow(zec)-n):nrow(zec),]
maid = maid[(nrow(maid)-n):nrow(maid),]

btc$ret = c(NA, diff(log(btc$btc.csv.close)))
dash$ret = c(NA, diff(log(dash$dash.csv.close)))
eth$ret = c(NA, diff(log(eth$eth.csv.close)))
iota$ret = c(NA, diff(log(iota$iot.csv.close)))
lsk$ret = c(NA, diff(log(lsk$lsk.csv.close)))
ltc$ret = c(NA, diff(log(ltc$ltc.csv.close)))
neo$ret = c(NA, diff(log(neo$neo.csv.close)))
xem$ret = c(NA, diff(log(xem$xem.csv.close)))
xmr$ret = c(NA, diff(log(xmr$xmr.csv.close)))
xrp$ret = c(NA, diff(log(xrp$xrp.csv.close)))
zec$ret = c(NA, diff(log(zec$zec.csv.close)))
maid$ret = c(NA, diff(log(maid$maid.csv.close)))

btc = btc[2:nrow(btc), ]
eth = eth[2:nrow(eth), ]
dash = dash[2:197, ]
iota = iota[2:197, ]
lsk = lsk[2:197, ]
ltc = ltc[2:197, ]
neo = neo[2:197, ]
xem = xem[2:197, ]
xmr = xmr[2:197, ]
xrp = xrp[2:197, ]
zec = zec[2:197, ]
maid = maid[2:197, ]

returns = cbind(btc$ret, eth$ret, dash$ret, iota$ret, lsk$ret, neo$ret
, xem$ret, xmr$ret, xrp$ret, zec$ret, maid$ret)
returns = data.frame(returns)
colnames(returns) = c("btc", "eth", "dash", "iota", "lsk", "neo", "xem
", "xmr", "xrp", "zec", "maid")

cor_mat = cor(returns)

#install.packages("corrplot")
library(corrplot)

## corrplot 0.84 loaded

corrplot(cor_mat, method='shade', type='full', shade.col=NA, tl.col='b
lack')
```

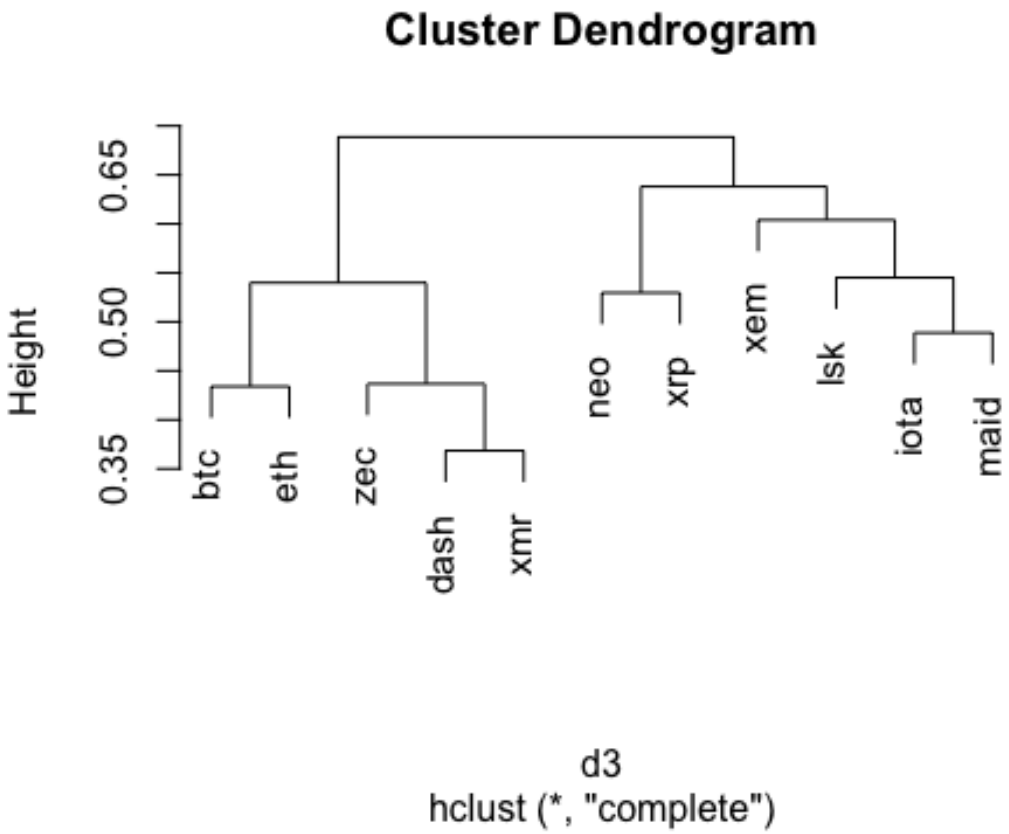



```
#install.packages("TSclust")
library(TSclust)

## Loading required package: wmtsa

## Loading required package: pdc

## Loading required package: cluster

D1 <- diss(returns, "COR")
summary(D1)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7483  0.9707  1.0530  1.0914  1.1438  1.4930

average_return_btc = mean(btc$ret)

#install.packages("dplyr")
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

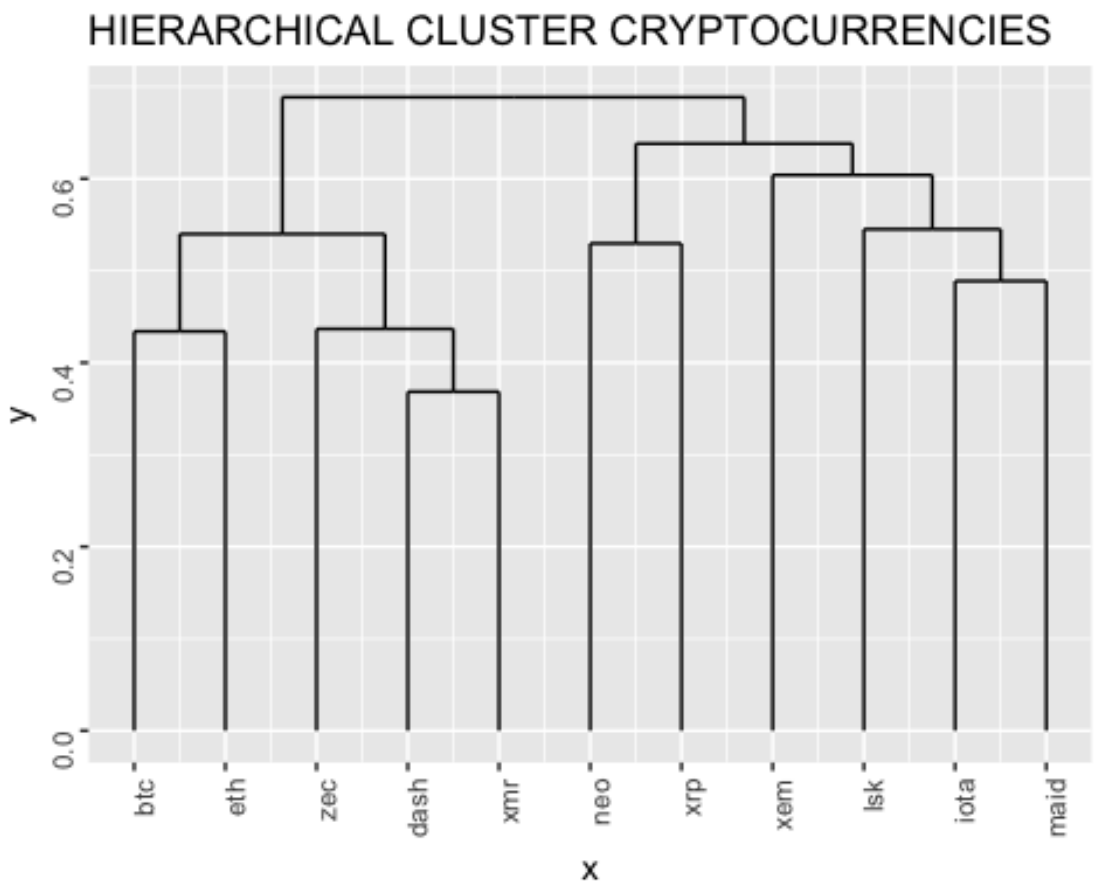
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

summary_ret = returns %>%
  summarize(av_ret_btc = mean(btc),
            av_ret_eth = mean(eth),
            av_ret_dash = mean(dash),
            av_ret_iota = mean(iota),
            av_ret_lsk = mean(lsk),
            av_ret_neo = mean(neo),
            av_ret_xem = mean(xem),
            av_ret_xmr = mean(xmr),
            av_ret_xrp = mean(xrp),
            av_ret_zec = mean(zec),
            av_ret_maid = mean(maid))
#here we see the average return for the period selected of the cryptoc
urrencies under consideration.

C1 <- hclust(D1)
C1

##
## Call:
## hclust(d = D1)
##
## Cluster method   : complete
## Number of objects: 11

plot(C1)
```




```
#D2 <- diss(returns, "FRECHET")
#the calculation takes too long for nothing; it's not worth running it
.

#c2 = hclust(D2)
#c2
#plot(c2)

#we are going to try more methods

d3 <- diss(returns, "ACF")

c3 = hclust(d3)
c3

##
## Call:
## hclust(d = d3)
##
## Cluster method   : complete
## Number of objects: 11

plot(c3)

#install.packages("ggdendro")
library(ggplot2)
```




```
library(ggdendro)

ggdendrogram(c3, rotate = FALSE, theme_dendro = FALSE, segments = TRUE
, labels = TRUE )  +
  ggtitle('HIERARCHICAL CLUSTER CRYPTOCURRENCIES')
```


HIERARCHICAL CLUSTER CRYPTOCURRENCIES
y
0.0
0.2
0.4
0.6
btc
eth
zec
dash
xmr
neo
xrp
xem
lsk
iota
maid
x

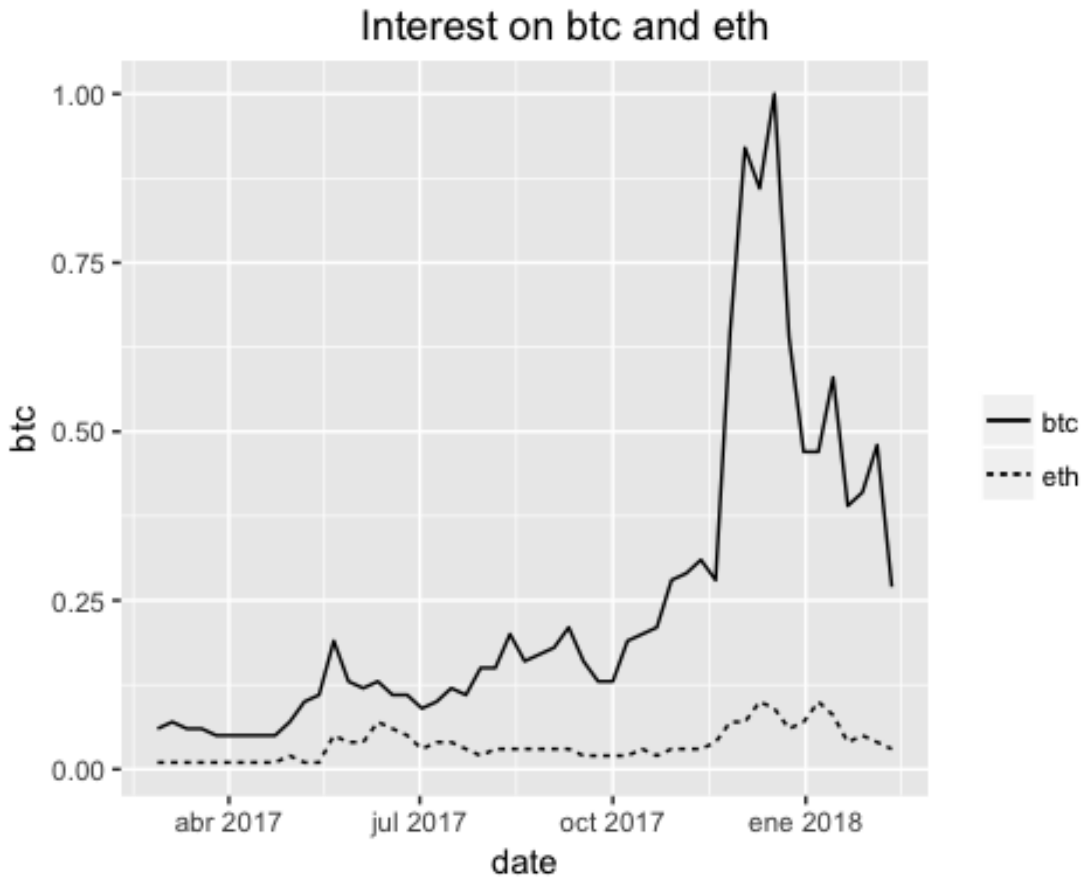
In the first chunk, the whole data mining and transformation process is performed, using the same code as in the previous part of Annex 1. However, as the directory must be changed. This is because in Annex 1, as seen in previous code, a generic function to read all csv is used. In this section, we need to read also, but separately, all data from Google Trends. This means that if we do not choose a new directory with all the cryptocurrencies' csvs plus the google trends csvs, our generic function would read the latter also; therefore the data for the beginning of Annex 1 would not work. For this reason, the first thing to do for running this Rmd is to choose as a directory (in my computer): ~Documents/TFG/google_interest_portfolio_selection. For the first part of the Annex 1 (the Rmd before this), the directory: ~Documents/TFG/portfolio_analysis_tfg should be chosen.

```
library(readr)
btc <- read_csv('btc.csv')

dash <- read_csv('dash.csv')

eth <- read_csv('eth.csv')

iota <- read_csv('iot.csv')

lsk <- read_csv('lsk.csv')

maid <- read_csv('maid.csv')

neo <- read_csv('neo.csv')

xem <- read_csv('xem.csv')

xmr <- read_csv('xmr.csv')

xrp <- read_csv('xrp.csv')

zec <- read_csv('zec.csv')

btc = data.frame(btc)
dash = data.frame(dash)
eth = data.frame(eth)
iota = data.frame(iota)
lsk = data.frame(lsk)
maid = data.frame(maid)
neo = data.frame(neo)
xem = data.frame(xem)
xmr = data.frame(xmr)
xrp = data.frame(xrp)
zec = data.frame(zec)


btc = btc[ , 2:5]
dash = dash[ , 2:5]
```

```
eth = eth[ , 2:5]
iota = iota[ , 2:5]
lsk = lsk[ , 2:5]
neo = neo[ , 2:5]
xem = xem[ , 2:5]
xmr = xmr[ , 2:5]
xrp = xrp[ , 2:5]
zec = zec[ , 2:5]
maid = maid[ , 2:5]


btc$volatility = btc[,3] - btc[,4]
dash$volatility = dash[,3] - dash[,4]
#we cannot use eos because there is no data about its volatility (high
-low)
eth$volatility = eth[,3] - eth[,4]
iota$volatility = iota[,3] - iota[,4]
lsk$volatility = lsk[,3] - lsk[,4]
neo$volatility = neo[,3] - neo[,4]
xem$volatility = xem[,3] - xem[,4]
xmr$volatility = xmr[,3] - xmr[,4]
xrp$volatility = xrp[,3] - xrp[,4]
zec$volatility = zec[,3] - zec[,4]
maid$volatility = maid[,3] - maid[,4]


n = 196

btc = btc[(nrow(btc)-n):nrow(btc),]
dash = dash[(nrow(dash)-n):nrow(dash),]
eth = eth[(nrow(eth)-n):nrow(eth),]
iota = iota[(nrow(iota)-n):nrow(iota),]
lsk = lsk[(nrow(lsk)-n):nrow(lsk),]
xem = xem[(nrow(xem)-n):nrow(xem),]
xmr = xmr[(nrow(xmr)-n):nrow(xmr),]
xrp = xrp[(nrow(xrp)-n):nrow(xrp),]
zec = zec[(nrow(zec)-n):nrow(zec),]
maid = maid[(nrow(maid)-n):nrow(maid),]

btc$ret = c(NA, diff(log(btc$close)))
dash$ret = c(NA, diff(log(dash$close)))
eth$ret = c(NA, diff(log(eth$close)))
iota$ret = c(NA, diff(log(iota$close)))
lsk$ret = c(NA, diff(log(lsk$close)))
neo$ret = c(NA, diff(log(neo$close)))
xem$ret = c(NA, diff(log(xem$close)))
xmr$ret = c(NA, diff(log(xmr$close)))
xrp$ret = c(NA, diff(log(xrp$close)))
zec$ret = c(NA, diff(log(zec$close)))
maid$ret = c(NA, diff(log(maid$close)))

btc = btc[2:nrow(btc), ]
eth = eth[2:nrow(eth), ]
dash = dash[2:197, ]
```

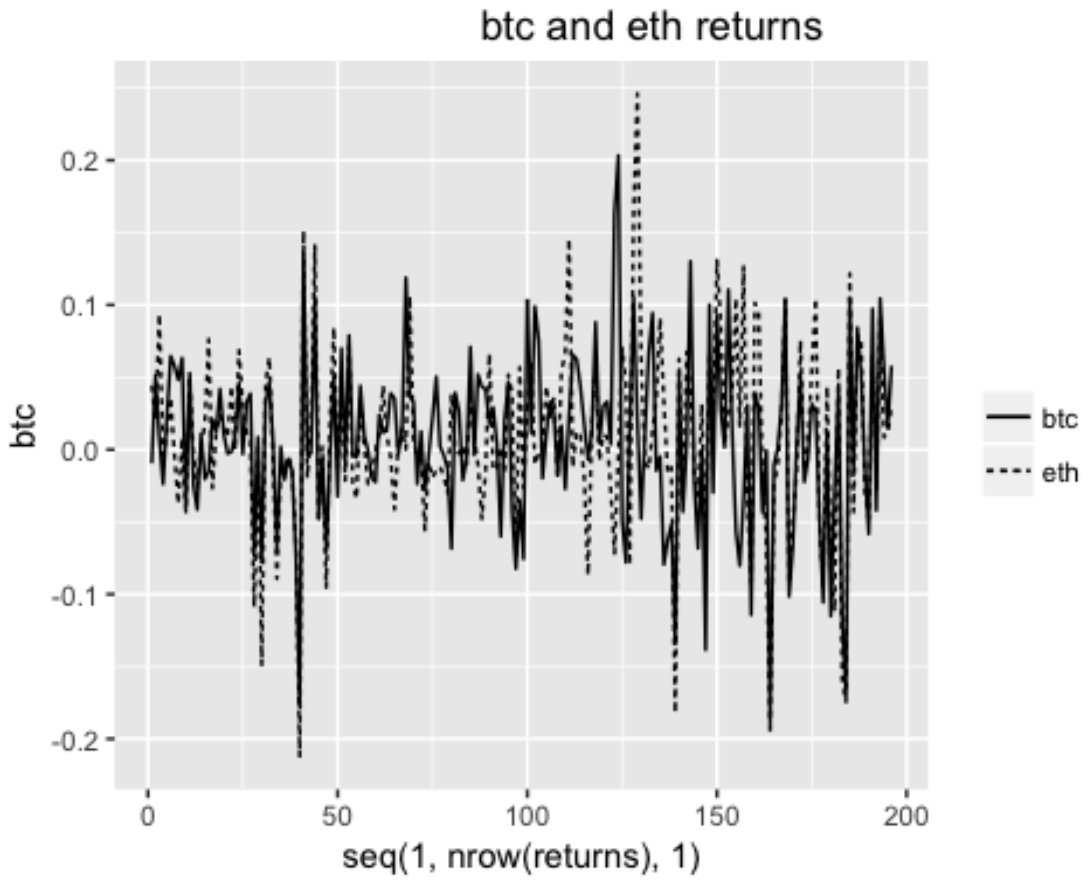
```
iota = iota[2:197, ]
lsk = lsk[2:197, ]
neo = neo[2:197, ]
xem = xem[2:197, ]
xmr = xmr[2:197, ]
xrp = xrp[2:197, ]
zec = zec[2:197, ]
maid = maid[2:197, ]

#the next step is to create tables for each category of cryptocurrenci
es, and put some variables

#for that we need to read the variables again, one by one, as done in
the previous code.


total_perc_ret_btc = (btc$close[nrow(btc)] - btc$close[1])/btc$close[1
]
print(total_perc_ret_btc)

## [1] 2.340743

total_perc_ret_eth = (eth$close[nrow(eth)] - eth$close[1])/eth$close[1
]
print(total_perc_ret_eth)

## [1] 2.655352

mean_daily_vol_btc = mean(btc$volatility)
print(mean_daily_vol_btc)

## [1] 863.044

mean_daily_vol_eth = mean(eth$volatility)
print(mean_daily_vol_eth) #much lower volatility

## [1] 59.27429

mean_price_btc = mean(btc$close)
mean_vol_to_mean_price_btc = mean_daily_vol_btc/mean_price_btc

mean_price_eth = mean(eth$close)
mean_vol_to_mean_price_eth = mean_daily_vol_eth/mean_price_eth

mean_daily_ret_btc = mean(btc$ret)
mean_daily_ret_eth = mean(eth$ret)

mean_daily_perc_vol_btc = mean(btc$volatility) / mean(btc$close)
mean_daily_perc_vol_eth = mean(eth$volatility) / mean(eth$close)


#we now need to take into account the google interest towards the cryp
tocurrencies

library(readr)
interest = read_csv('eth_btc_google.csv')
```

```
## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   btc = col_integer(),
##   eth = col_integer()
## )

interest = data.frame(interest)
interest$btc = interest$btc/100
interest$eth = interest$eth/100
interest$date = as.Date(interest$date)

library(ggplot2)

btc_eth_int <- ggplot(data = interest, aes(x = date)) +
  geom_line(aes(y = btc, linetype = "btc")) +
  geom_line(aes(y = eth, linetype = "eth")) +
  ggtitle('                    Interest on btc and eth      ') +
  scale_linetype_discrete(name ="")
btc_eth_int
```




```
returns = cbind(btc$ret, eth$ret, dash$ret, iota$ret, lsk$ret, neo$ret
, xem$ret, xmr$ret, xrp$ret, zec$ret, maid$ret)
returns = data.frame(returns)
colnames(returns) = c("btc", "eth", "dash", "iota", "lsk", "neo", "xem
", "xmr", "xrp", "zec", "maid")
```

```r
avg_interest_btc = mean(interest$btc)
avg_interest_eth = mean(interest$eth)
btc2 = read_csv('btc.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

nrow(btc2) #this number of observations is going to be used for taking the inverse of the time, so that we give more weight to currencies tha t have had good performance but are not still in the maturity stage, n ot even close (btc is also not in the maturity stage, but we expect et hereum to have more relevance in the coming years, as in less time it has done as much as bitcoin for the blockchain community, if not more)

## [1] 2773

inv_time_btc = 1 / (nrow(btc2) / 365) #0.1316264

eth2 = read_csv('eth.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

nrow(eth2)

## [1] 926

inv_time_eth = 1 / (nrow(eth2) / 365) #0.3941685

#let's visualize the movement of the returns of both series
ggplot(data = returns, aes(x = seq(1, nrow(returns), 1))) +
  geom_line(aes(y = btc, linetype = "btc")) +
  geom_line(aes(y = eth, linetype = "eth")) +
  ggtitle('                                btc and eth returns') +
  scale_linetype_discrete(name = "")
```




```
#this chunk contains all the rest of calculations for including in the
excel spreadsheet (where the final computation of portfolio weights is
carried out)

#anonymity cryptos
total_perc_ret_xmr = (xmr$close[nrow(xmr)] - xmr$close[1])/xmr$close[1
]
total_perc_ret_zec = (zec$close[nrow(zec)] - zec$close[1])/zec$close[1
]
total_perc_ret_dash = (dash$close[nrow(dash)] - dash$close[1])/dash$cl
ose[1]

mean_ret_xmr = mean(returns$xmr)
mean_ret_zec = mean(returns$zec)
mean_ret_dash = mean(returns$dash)


anonim_interest = read_csv('anonimity_cryptos_interest.csv')

anonim_interest = data.frame(anonim_interest)
anonim_interest$dash = anonim_interest$dash/100
anonim_interest$zec = anonim_interest$zec/100
anonim_interest$xmr = anonim_interest$xmr/100

mean_int_dash = mean(anonim_interest$dash)
mean_int_zec = mean(anonim_interest$zec)
mean_int_xmr = mean(anonim_interest$xmr)
```

```r
mean_vol_dash = mean(dash$volatility)
mean_vol_zec = mean(zec$volatility)
mean_vol_xmr = mean(xmr$volatility)

mean_perc_vol_dash = mean_vol_dash / mean(dash$close)
mean_perc_vol_zec = mean_vol_zec / mean(zec$close)
mean_perc_vol_xmr = mean_vol_xmr / mean(xmr$close)

xmr2 = read_csv('xmr.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

nrow(xmr2)

## [1] 1116

inv_time_xmr = 1 / (nrow(xmr2) / 365)

zec2 = read_csv('zec.csv')

inv_time_zec = 1 / (nrow(zec2) / 365)

dash2 = read_csv('dash.csv')

inv_time_dash = 1 / (nrow(dash2) / 365)


xem2 = read_csv('xem.csv')

tot_perc_ret_xem = (xem$close[nrow(xem)] - xem$close[1]) / xem$close[1
]
tot_perc_ret_xem

## [1] 1.402948

xrp2 = read_csv('xrp.csv')

tot_perc_ret_xrp = (xrp$close[nrow(xrp)] - xrp$close[1] ) / xrp$close[
1]
tot_perc_ret_xrp

## [1] 5.437292

tot_perc_ret_neo = (neo$close[nrow(neo)] - neo$close[1]) / neo$close[1
]
tot_perc_ret_neo
```

```
## [1] 7.43707

mean_daily_ret_xem = mean(xem$ret)
mean_daily_ret_xem

## [1] 0.004787476

mean_daily_ret_xrp = mean(xrp$ret)
mean_daily_ret_xrp

## [1] 0.009371945

mean_daily_ret_neo = mean(neo$ret)
mean_daily_ret_neo

## [1] 0.01150895

platform_int = read_csv('platform_interest.csv')

## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   xem = col_integer(),
##   xrp = col_character(),
##   neo = col_integer()
## )

platform_int = data.frame(platform_int)
platform_int$xrp[1:3] = c(0,0,0) #we clean the data to have only numbe
rs.
platform_int$xem = platform_int$xem/100
platform_int$xrp = as.numeric(platform_int$xrp)
#platform_int = gsub(pattern = '<1', x = platform_int, replacement = 0
.5)
platform_int$xrp = platform_int$xrp/100
platform_int$neo = platform_int$neo/100

mean_vol_xem = mean(xem$volatility)
mean_vol_xrp = mean(xrp$volatility)
mean_vol_neo = mean(neo$volatility)
mean_vol_xem

## [1] 0.09538571

mean_vol_xrp

## [1] 0.1119

mean_vol_neo

## [1] 10.43291

mean_perc_vol_xem = mean_vol_xem/mean(xem$close)
mean_perc_vol_xrp = mean_vol_xrp/mean(xrp$close)
mean_perc_vol_neo = mean_vol_neo/mean(neo$close)

inv_time_xem = 1/(nrow(xem2)/365)
```

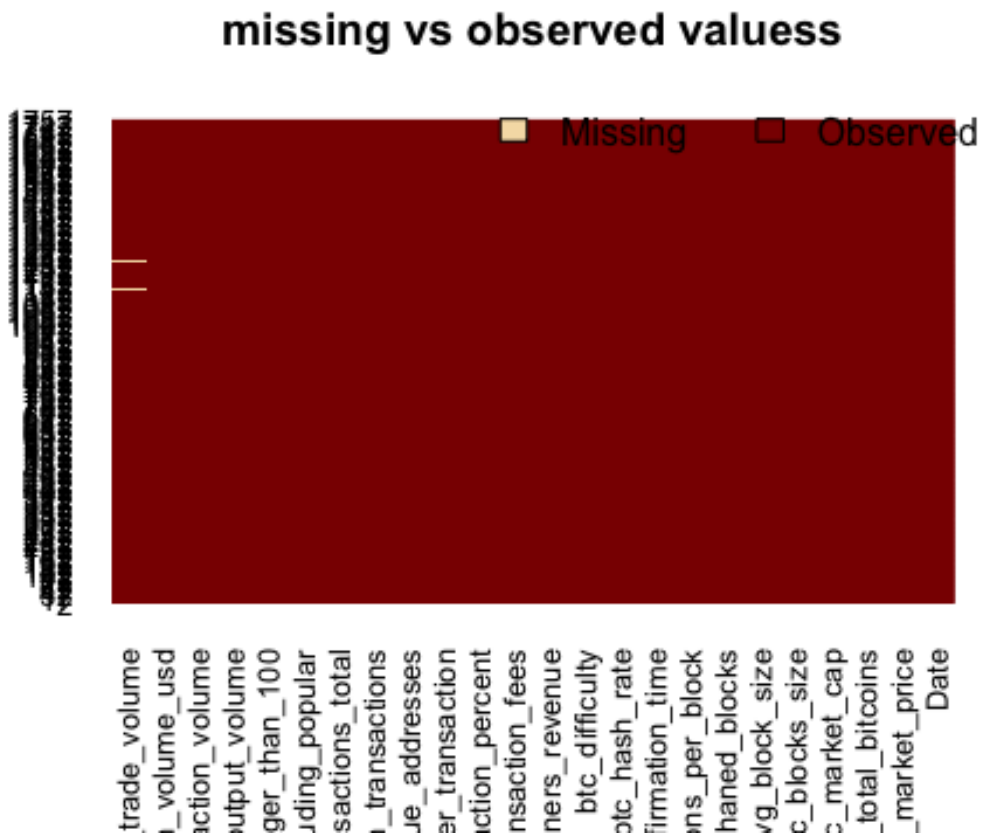
```r
inv_time_xrp = 1/(nrow(xrp2)/365)

neo2 = read_csv('neo.csv')

inv_time_neo = 1/(nrow(neo2)/365)


#now let's do the protocol currencies analysis
tot_perc_ret_iota = (iota$close[nrow(iota)] - iota$close[1]) / iota$cl
ose[1]
tot_perc_ret_lsk = (lsk$close[nrow(lsk)] - lsk$close[1]) / lsk$close[1
]
tot_perc_ret_maid = (maid$close[nrow(maid)] - maid$close[1]) / maid$cl
ose[1]

mean_daily_ret_iota = mean(iota$ret)
mean_daily_ret_iota

## [1] 0.008573417

mean_daily_ret_lsk = mean(lsk$ret)
mean_daily_ret_lsk

## [1] 0.01424316

mean_daily_ret_maid = mean(maid$ret)
mean_daily_ret_maid

## [1] 0.001079182

#now let us check the google interest in each of the protocol currenci
es

protocol_int = read_csv('protocol_interest.csv')

protocol_int$lsk = ifelse(protocol_int$lsk == "<1", 0.5, protocol_int$
lsk) #we transform all values <1 to 0.5, as the expected mean between
0 and 1 is supposed to be this if the range behaves normally (bell-sha
ped).
protocol_int$maid = ifelse(protocol_int$maid == "<1", 0.5, protocol_in
t$maid)

protocol_int$iota = as.numeric(protocol_int$iota)
protocol_int$lsk = as.numeric(protocol_int$lsk)
protocol_int$maid = as.numeric(protocol_int$maid)

protocol_int$iota = protocol_int$iota/100
protocol_int$lsk = protocol_int$lsk/100
protocol_int$maid = protocol_int$maid / 100

avg_int_iota = mean(protocol_int$iota)
avg_int_lsk = mean(protocol_int$lsk)
avg_int_maid = mean(protocol_int$maid)

mean_vol_iota = mean(iota$volatility)
mean_vol_lsk = mean(lsk$volatility)
```

```r
mean_vol_maid = mean(maid$volatility)

mean_vol_iota

## [1] 0.3354153

mean_vol_lsk

## [1] 2.117857

mean_vol_maid

## [1] 0.09970969

perc_vol_iota = mean_vol_iota / mean(iota$close)
perc_vol_lsk = mean_vol_lsk / mean(lsk$close)
perc_vol_maid = mean_vol_maid / mean(maid$close)

perc_vol_iota

## [1] 0.2063972

perc_vol_lsk

## [1] 0.1786537

perc_vol_maid

## [1] 0.1829106

iota2 = read_csv('iot.csv')

inv_time_iota = 1 / (nrow(iota2) / 365)

lsk2 = read_csv('lsk.csv')

inv_time_lsk = 1 / (nrow(lsk2) / 365)

maid2 = read_csv('maid.csv')

inv_time_maid = 1 / (nrow(maid) / 365)

inv_time_iota

## [1] 1.46

inv_time_lsk

## [1] 1.01108

inv_time_maid

## [1] 1.862245
```

ANNEX 2

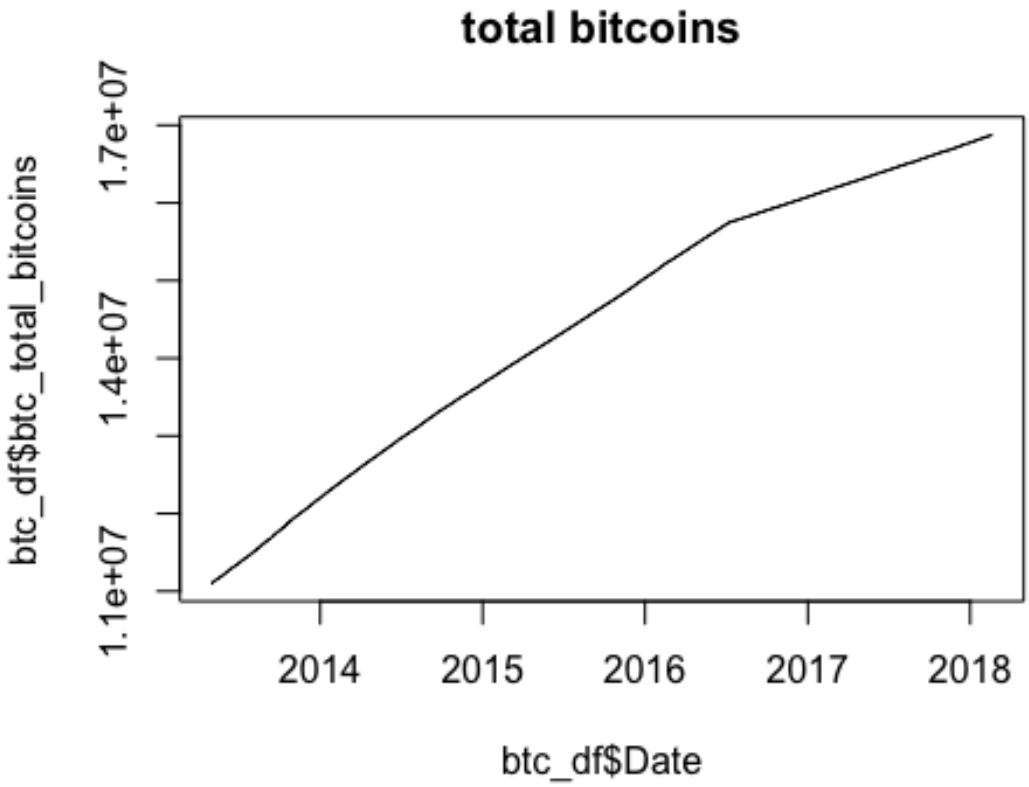
```r
library(readr)
btc_df <- read_csv('bitcoin_dataset.csv')

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Date = col_datetime(format = "")
## )

## See spec(...) for full column specifications.

btc_df$Date = as.Date(btc_df$Date)
class(btc_df$Date)

## [1] "Date"

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#install.packages("leaps")
library(leaps)

btc_df <- btc_df %>%
  filter(Date  >= '2013-05-01')

row_missing = c()
col_missing = c()

for(i in 1:nrow(btc_df))
  for (j in 2:ncol(btc_df)) {
    if (is.na(btc_df[i, j]) == TRUE | btc_df[i, j] == 0) {
      row_missing <- c(row_missing, i)
      col_missing <- c(col_missing, j)
      print(btc_df[i, j])
    }
  } #we use this loop to detect the columns and rows with missing valu
es

#we hope that NAs and 0s will concentrate in certain variables; which
we should eliminate.

col_missing_factor <- as.factor(col_missing)
levels(col_missing_factor) #it seems that only cols 5 and 8 have missi
ng values or 0.
```

```
## [1] "5" "8"

#these are 'btc_trade_vol' and 'btc_n_orphaned_blocks'.
#the latter will be eliminated completely,
#but for the former the mean of the last 20 days of trading volume wil
l be used for filling the missing values.

#install.packages("Amelia")
require(Amelia)

## Loading required package: Amelia

## Loading required package: Rcpp

## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.4, built: 2015-12-05)
## ## Copyright (C) 2005-2018 James Honaker, Gary King and Matthew Bla
ckwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##

missmap(btc_df, main = 'missing vs observed valuess')

## Warning in if (class(obj) == "amelia") {: la condición tiene longit
ud > 1 y
## sólo el primer elemento será usado

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'arguments'.

## Warning: Unknown or uninitialised column: 'imputations'.
```

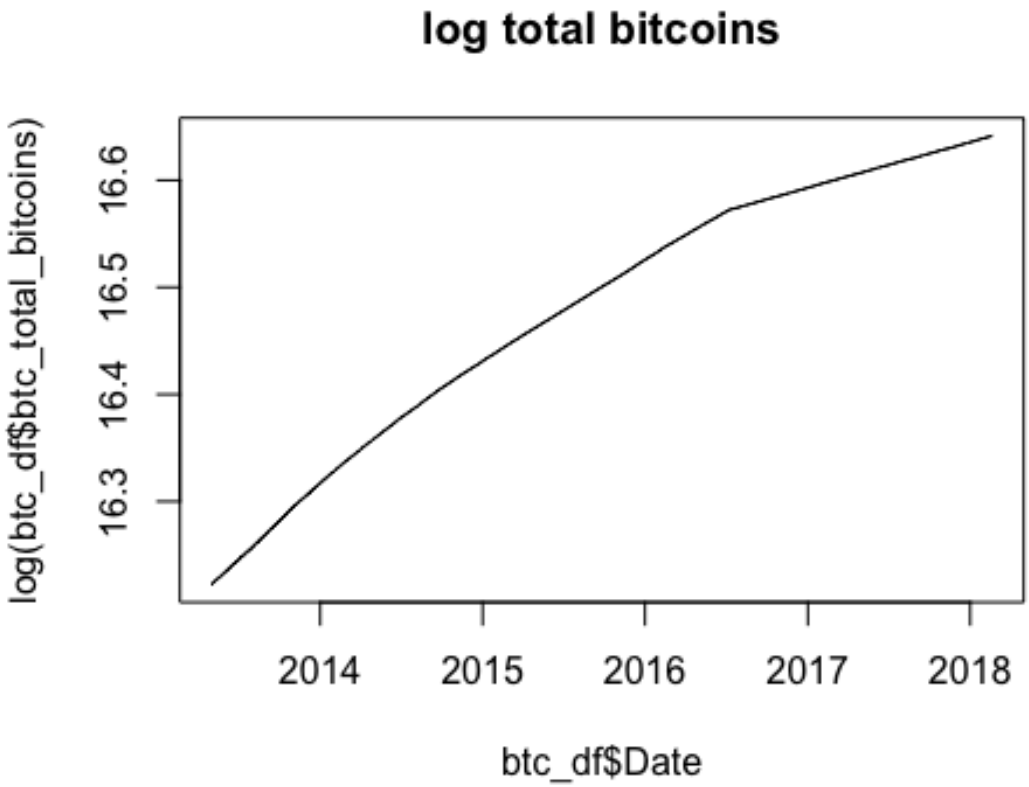



```
#install.packages("imputeTS")
library(imputeTS) #this package provides methods for imputing the miss
ing values in a time series framework

#I will use the method of weighted moving average with a window of 4 d
ays and the weighting will be exponential.
btc_df$btc_trade_volume <- na.ma(btc_df$btc_trade_volume,
                                 k = 5, weighting = 'exponential')

btc_df <- btc_df[, -8]
#we have to make some other transformations to avoid magnitude biases
as well as multicollinearity
colnames(btc_df)

##  [1] "Date"
##  [2] "btc_market_price"
##  [3] "btc_total_bitcoins"
##  [4] "btc_market_cap"
##  [5] "btc_trade_volume"
##  [6] "btc_blocks_size"
##  [7] "btc_avg_block_size"
##  [8] "btc_n_transactions_per_block"
##  [9] "btc_median_confirmation_time"
## [10] "btc_hash_rate"
## [11] "btc_difficulty"
## [12] "btc_miners_revenue"
## [13] "btc_transaction_fees"
```

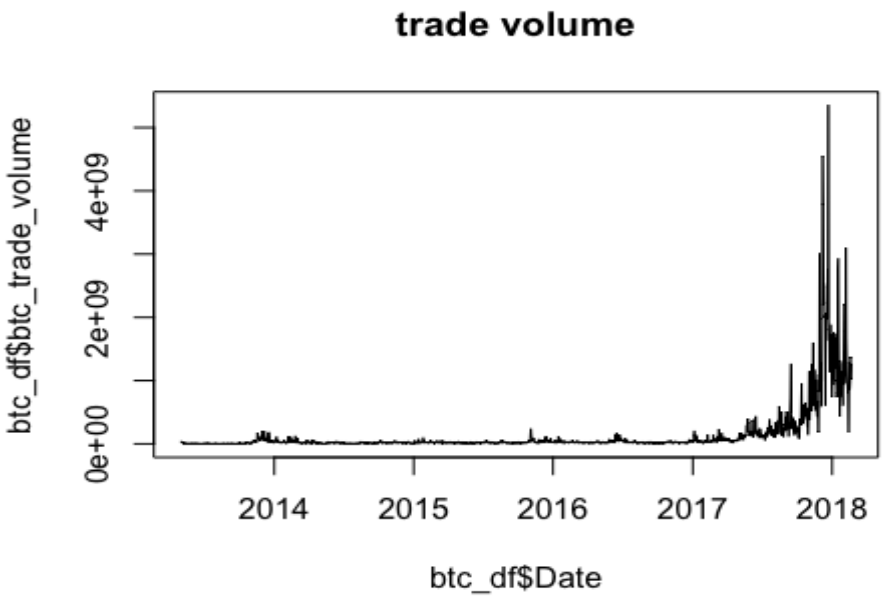
```
## [14] "btc_cost_per_transaction_percent"
## [15] "btc_cost_per_transaction"
## [16] "btc_n_unique_addresses"
## [17] "btc_n_transactions"
## [18] "btc_n_transactions_total"
## [19] "btc_n_transactions_excluding_popular"
## [20] "btc_n_transactions_excluding_chains_longer_than_100"
## [21] "btc_output_volume"
## [22] "btc_estimated_transaction_volume"
## [23] "btc_estimated_transaction_volume_usd"

btc_df <- btc_df[, -4]
btc_df <- btc_df[ , -22]
btc_df$ret <- c(NA, diff(log(btc_df$btc_market_price)))

#let's do some plots of certain variables...
plot(x = btc_df$Date, y = btc_df$btc_total_bitcoins, type = "l", main
= "total bitcoins")
```



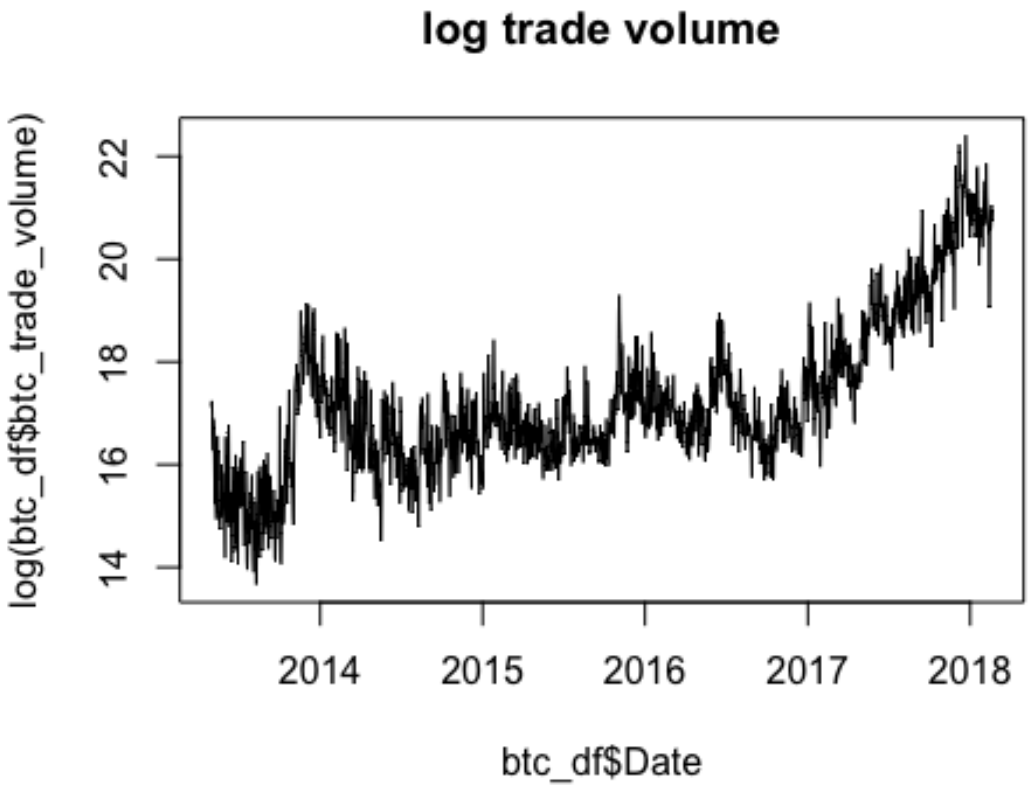
```r
plot(x = btc_df$Date, y = log(btc_df$btc_total_bitcoins), type = "l",
main = "log total bitcoins")
```



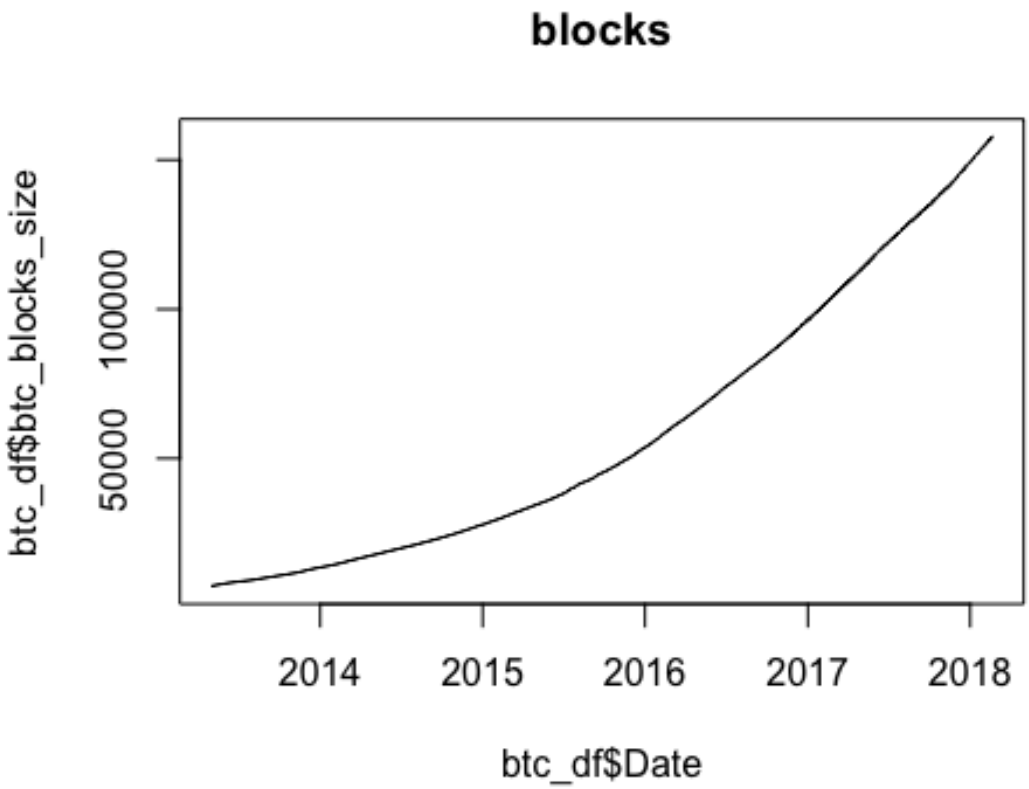

```r
plot(x = btc_df$Date, y = btc_df$btc_trade_volume, type = "l", main =
'trade volume' )
```



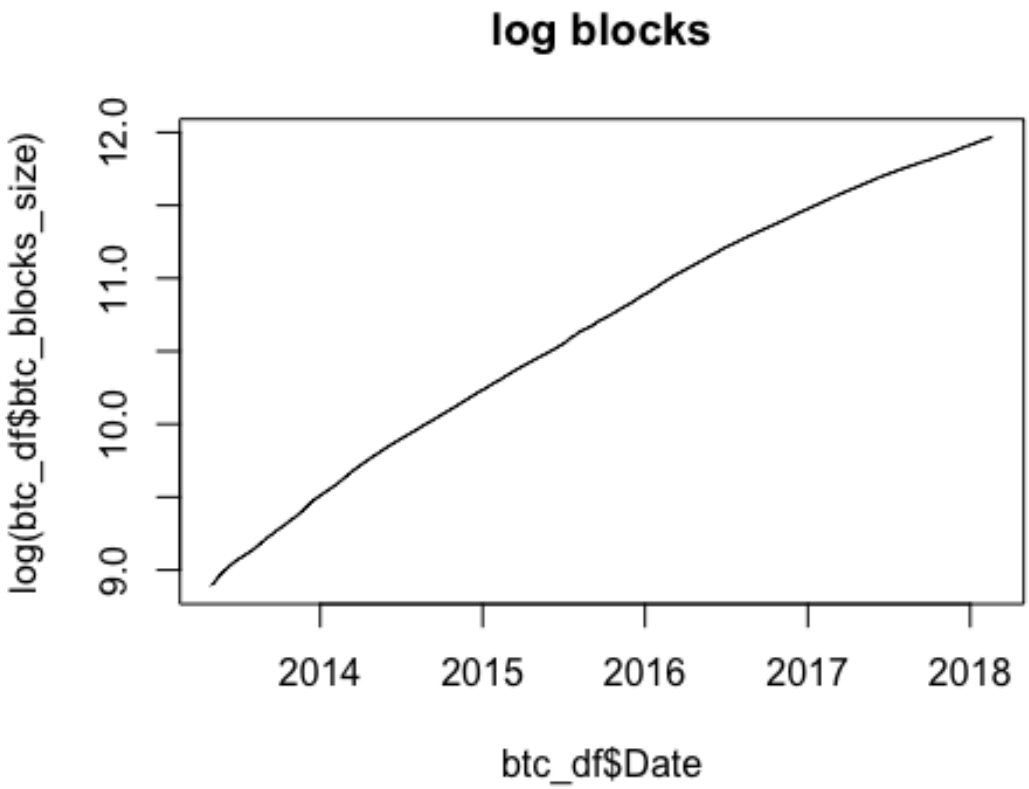
```
plot(x = btc_df$Date, y = log(btc_df$btc_trade_volume), type = "l", ma
in = 'log trade volume')
```



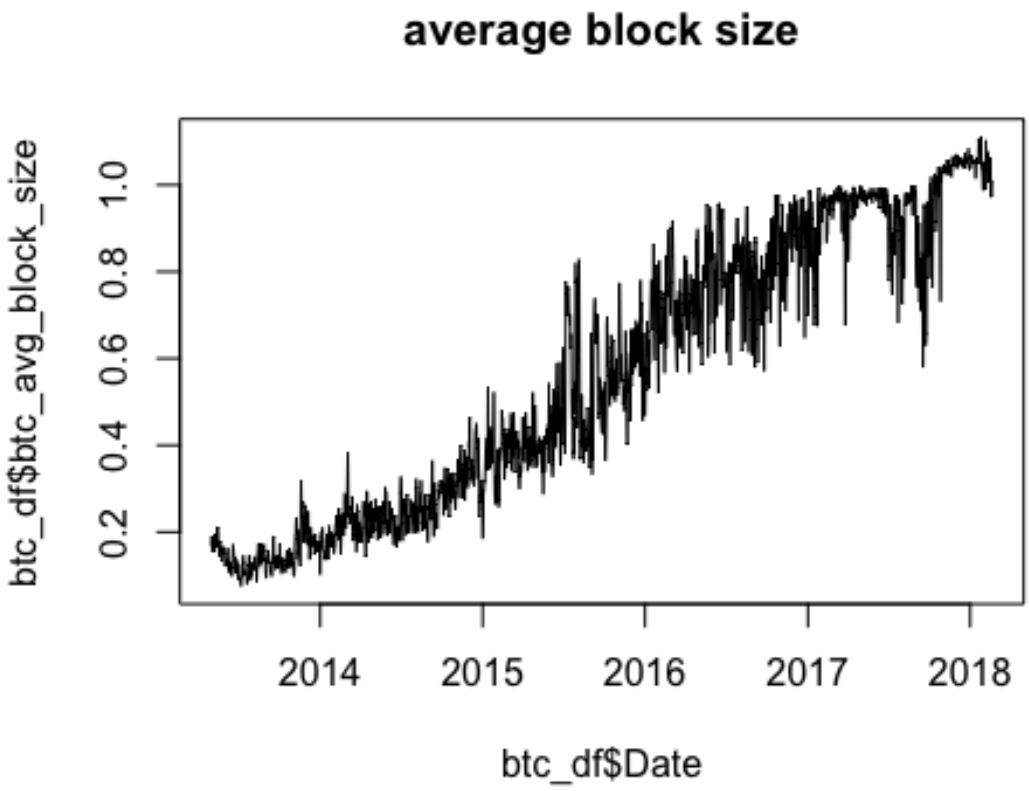

```
plot(x = btc_df$Date, y = btc_df$btc_blocks_size, type = "l", main = '
blocks')
```



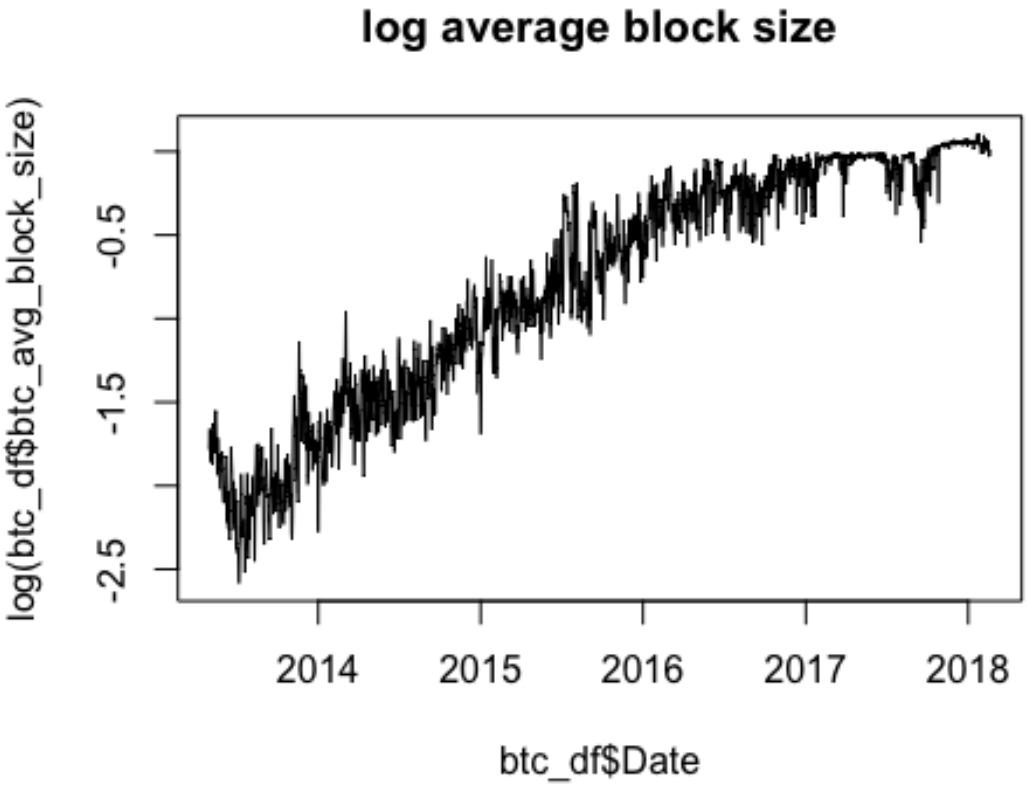

```
plot(x = btc_df$Date, y = log(btc_df$btc_blocks_size), type = "l", mai
n = "log blocks")
```




```
plot(x = btc_df$Date, y = btc_df$btc_avg_block_size, type = "l", main
= 'average block size')
```




```
plot(x = btc_df$Date, y = log(btc_df$btc_avg_block_size), type = "l",
main = 'log average block size')
```



The variables used by Majan, Saluja and Zhao (w.d.):

Average Confirmation Time	Ave. time to accept transaction in block
Block Size	Average block size in MB
Cost per transaction percent	Miners revenue divided by the number of transactions
Difficulty	How difficult it is to find a new block
Estimated Transaction Volume	Total output volume without change from value
Hash Rate	Bitcoin network giga hashes per second
Market Capitalization	Number of Bitcoins in circulation * the market price

Miners Revenue	(number of BTC mined/day * market price) + transaction fees
Number of Orphaned Blocks	Number of blocks mined / day not off blockchain
Number of TXN per block	Average number of transactions per block
Number of TXN	Total number of unique Bitcoin transactions per day
Number of unique addresses	Number of unique Bitcoin addresses used per day
Total Bitcoins Historical	total Number of Bitcoins mined
TXN Fees	Total BTC value of transaction fees miners earn/day
Trade Volume	USD trade volume from the top exchanges
Transaction to trade ratio	Relationship of BTC transaction volume and USD volume

They use the differential for each of these variables.

```
library(readr)
btc_df <- read_csv('bitcoin_dataset.csv')

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Date = col_datetime(format = "")
## )

## See spec(...) for full column specifications.

btc_df$Date = as.Date(btc_df$Date)
class(btc_df$Date)

## [1] "Date"

library(dplyr)
#install.packages("leaps")
library(leaps)

btc_df <- btc_df %>%
  filter(Date  >= '2013-05-01')
```

```r
btc_df$n_btc <- btc_df$btc_market_cap / btc_df$btc_market_price #creat
e variable number of transactions
btc_df$trans_to_trade <-
  btc_df$btc_estimated_transaction_volume /     btc_df$btc_estimated_t
ransaction_volume_usd #create variable transactions to trade described
by Majan, Saluja and Zhao (w.d)

btc_df <- btc_df[, c(1,2,14,10,5,7,9,17,13,15,11,12,22,18,25,26)]
library(imputeTS) #this package provides methods for imputing the miss
ing values in a time series framework

#we will use the method of weighted moving average with a window of 4
days and the weighting will be exponential.
btc_df$btc_trade_volume <- na.ma(btc_df$btc_trade_volume,
                                 k = 5, weighting = 'exponential')

diflog <- function (x) {
  x <- as.numeric(x)
  x <- diff(log(x))
}

diff_btc <- apply(btc_df[ ,2:ncol(btc_df)], MARGIN = 2, FUN = diflog)
#the variable difficulty is almost always 0; therefore we will take it
out.
diff_btc <- diff_btc[, -11]
diff_btc <- data.frame(diff_btc)

train_index <- seq(1, 0.8*nrow(diff_btc), 1)
#train = diff_btc[1:0.7*nrow(diff_btc), ]
btc_train <- diff_btc[train_index, ]
btc_test <- diff_btc[-train_index, ]

nullmodel = lm(btc_market_price ~1, data = btc_train)
fullmodel = lm(btc_market_price ~., data = btc_train)

model.step=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel
), direction='both')
```

```
## Start:  AIC=-9062.36
## btc_market_price ~ 1
##
##                                    Df Sum of Sq     RSS      AIC
## + trans_to_trade                    1   2.14896 0.05662 -14202.5
## + btc_miners_revenue                1   0.19072 2.01485  -9187.3
## + n_btc                             1   0.02169 2.18388  -9074.2
## + btc_avg_block_size                1   0.01799 2.18759  -9071.9
## + btc_n_transactions                1   0.01015 2.19542  -9066.8
## + btc_transaction_fees              1   0.00668 2.19889  -9064.6
## + btc_n_unique_addresses            1   0.00429 2.20128  -9063.1
## + btc_cost_per_transaction_percent  1   0.00332 2.20225  -9062.5
## <none>                                          2.20557  -9062.4
## + btc_hash_rate                     1   0.00133 2.20424  -9061.2
## + btc_trade_volume                  1   0.00112 2.20445  -9061.1
```

```
## + btc_median_confirmation_time      1   0.00051 2.20506  -9060.7
## + btc_output_volume                 1   0.00033 2.20524  -9060.6
## + btc_n_transactions_per_block      1   0.00000 2.20557  -9060.4
##
## Step:  AIC=-14202.45
## btc_market_price ~ trans_to_trade
##
##                                   Df Sum of Sq     RSS      AIC
## + n_btc                            1   0.05660 0.00001 -26180.7
## + btc_output_volume                1   0.00023 0.05638 -14206.2
## + btc_cost_per_transaction_percent  1   0.00018 0.05644 -14204.8
## <none>                                         0.05662 -14202.5
## + btc_n_transactions_per_block      1   0.00005 0.05657 -14201.6
## + btc_n_transactions                1   0.00004 0.05658 -14201.4
## + btc_hash_rate                     1   0.00001 0.05660 -14200.8
## + btc_median_confirmation_time      1   0.00001 0.05660 -14200.8
## + btc_trade_volume                  1   0.00001 0.05660 -14200.7
## + btc_transaction_fees              1   0.00001 0.05661 -14200.7
## + btc_avg_block_size                1   0.00000 0.05661 -14200.5
## + btc_n_unique_addresses            1   0.00000 0.05661 -14200.5
## + btc_miners_revenue                1   0.00000 0.05661 -14200.5
## - trans_to_trade                    1   2.14896 2.20557  -9062.4
##
## Step:  AIC=-26180.67
## btc_market_price ~ trans_to_trade + n_btc
##
##                                   Df Sum of Sq     RSS      AIC
## + btc_hash_rate                    1    0.0000 0.00001 -26246.3
## + btc_miners_revenue               1    0.0000 0.00001 -26221.8
## + btc_avg_block_size               1    0.0000 0.00001 -26197.8
## + btc_median_confirmation_time     1    0.0000 0.00001 -26194.6
## + btc_cost_per_transaction_percent  1    0.0000 0.00001 -26183.0
## <none>                                        0.00001 -26180.7
## + btc_trade_volume                 1    0.0000 0.00001 -26179.2
## + btc_transaction_fees             1    0.0000 0.00001 -26178.9
## + btc_n_unique_addresses           1    0.0000 0.00001 -26178.9
## + btc_n_transactions_per_block     1    0.0000 0.00001 -26178.9
## + btc_n_transactions               1    0.0000 0.00001 -26178.7
## + btc_output_volume                1    0.0000 0.00001 -26178.7
## - n_btc                            1    0.0566 0.05662 -14202.5
## - trans_to_trade                   1    2.1839 2.18388  -9074.2
##
## Step:  AIC=-26246.27
## btc_market_price ~ trans_to_trade + n_btc + btc_hash_rate
##
##                                   Df Sum of Sq     RSS      AIC
## + btc_miners_revenue               1   0.00000 0.00001 -26255.6
## <none>                                         0.00001 -26246.3
## + btc_transaction_fees             1   0.00000 0.00001 -26245.4
## + btc_median_confirmation_time     1   0.00000 0.00001 -26245.2
## + btc_n_transactions               1   0.00000 0.00001 -26244.9
## + btc_n_transactions_per_block     1   0.00000 0.00001 -26244.8
## + btc_trade_volume                 1   0.00000 0.00001 -26244.3
## + btc_output_volume                1   0.00000 0.00001 -26244.3
```

```
## + btc_n_unique_addresses             1   0.00000 0.00001 -26244.3
## + btc_avg_block_size                 1   0.00000 0.00001 -26244.3
## + btc_cost_per_transaction_percent   1   0.00000 0.00001 -26244.3
## - btc_hash_rate                      1   0.00000 0.00001 -26180.7
## - n_btc                              1   0.05659 0.05660 -14200.8
## - trans_to_trade                     1   2.18238 2.18239  -9073.2
##
## Step:  AIC=-26255.62
## btc_market_price ~ trans_to_trade + n_btc + btc_hash_rate + btc_min
ers_revenue
##
##                                   Df Sum of Sq     RSS    AIC
## <none>                                         0.00001 -26256
## + btc_transaction_fees               1   0.00000 0.00001 -26254
## + btc_n_transactions_per_block       1   0.00000 0.00001 -26254
## + btc_avg_block_size                 1   0.00000 0.00001 -26254
## + btc_n_transactions                 1   0.00000 0.00001 -26254
## + btc_median_confirmation_time       1   0.00000 0.00001 -26254
## + btc_cost_per_transaction_percent   1   0.00000 0.00001 -26254
## + btc_n_unique_addresses             1   0.00000 0.00001 -26254
## + btc_trade_volume                   1   0.00000 0.00001 -26254
## + btc_output_volume                  1   0.00000 0.00001 -26254
## - btc_miners_revenue                 1   0.00000 0.00001 -26246
## - btc_hash_rate                      1   0.00000 0.00001 -26222
## - n_btc                              1   0.05655 0.05656 -14200
## - trans_to_trade                     1   1.11113 1.11114 -10019
```

```
summary(model.step)
```

```
##
## Call:
## lm(formula = btc_market_price ~ trans_to_trade + n_btc + btc_hash_r
ate +
##     btc_miners_revenue, data = btc_train)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.536e-04 -3.060e-05 -8.820e-07  5.731e-05  2.633e-04
##
## Coefficients:
##                      Estimate Std. Error    t value Pr(>|t|)
## (Intercept)         2.662e-04  2.368e-06    112.447  < 2e-16 ***
## trans_to_trade     -1.000e+00  8.235e-05 -12146.015  < 2e-16 ***
## n_btc              -9.997e-01  3.648e-04  -2740.120  < 2e-16 ***
## btc_hash_rate       3.513e-04  5.839e-05      6.016 2.28e-09 ***
## btc_miners_revenue -1.997e-04  5.926e-05     -3.369 0.000774 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.679e-05 on 1399 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.321e+07 on 4 and 1399 DF,  p-value: < 2.2e-16
```

A hypothesis is that trans_to_trade is actually 'cheating' the model, in the sense that this variable could intrinsically include the price. It is exceptionally good to have an R^2 of 1, and that makes the model look suspicious. If predicting the price with that accuracy is unthinkable, how can it be that with 4 variables the price variations can be explained? Looking at the trans_to_trade variable, it is clear that it is -1*price; therefore it is eliminated from the dataset in order to calculate the linear explanatory model again.

```
diff_btc <- diff_btc[ , -14]

train_index <- seq(1, 0.8*nrow(diff_btc), 1)
#train = diff_btc[1:0.7*nrow(diff_btc), ]
btc_train <- diff_btc[train_index, ]
btc_test <- diff_btc[-train_index, ]

nullmodel = lm(btc_market_price ~1, data = btc_train)
fullmodel = lm(btc_market_price ~., data = btc_train)

model.step=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel
), direction='both')

## Start:  AIC=-9062.36
## btc_market_price ~ 1
##
##                                    Df Sum of Sq    RSS     AIC
## + btc_miners_revenue                1  0.190718 2.0149 -9187.3
## + n_btc                             1  0.021689 2.1839 -9074.2
## + btc_avg_block_size                1  0.017985 2.1876 -9071.9
## + btc_n_transactions                1  0.010154 2.1954 -9066.8
## + btc_transaction_fees              1  0.006676 2.1989 -9064.6
## + btc_n_unique_addresses            1  0.004287 2.2013 -9063.1
## + btc_cost_per_transaction_percent  1  0.003323 2.2022 -9062.5
## <none>                                         2.2056 -9062.4
## + btc_hash_rate                     1  0.001334 2.2042 -9061.2
## + btc_trade_volume                  1  0.001123 2.2045 -9061.1
## + btc_median_confirmation_time      1  0.000509 2.2051 -9060.7
## + btc_output_volume                 1  0.000330 2.2052 -9060.6
## + btc_n_transactions_per_block      1  0.000000 2.2056 -9060.4
##
## Step:  AIC=-9187.33
## btc_market_price ~ btc_miners_revenue
##
##                                    Df Sum of Sq    RSS     AIC
## + btc_hash_rate                     1   0.85930 1.1556 -9965.9
## + btc_avg_block_size                1   0.17195 1.8429 -9310.6
## + btc_median_confirmation_time      1   0.10763 1.9072 -9262.4
## + n_btc                             1   0.02370 1.9911 -9201.9
## + btc_cost_per_transaction_percent  1   0.00410 2.0107 -9188.2
## + btc_n_transactions_per_block      1   0.00303 2.0118 -9187.4
## <none>                                         2.0149 -9187.3
## + btc_transaction_fees              1   0.00137 2.0135 -9186.3
## + btc_n_transactions                1   0.00106 2.0138 -9186.1
## + btc_output_volume                 1   0.00090 2.0139 -9186.0
## + btc_n_unique_addresses            1   0.00042 2.0144 -9185.6
## + btc_trade_volume                  1   0.00027 2.0146 -9185.5
```

```
## - btc_miners_revenue                1   0.19072 2.2056 -9062.4
##
## Step:  AIC=-9965.91
## btc_market_price ~ btc_miners_revenue + btc_hash_rate
##
##                                   Df Sum of Sq    RSS      AIC
## + n_btc                            1   0.04442 1.1111 -10018.9
## + btc_avg_block_size               1   0.03956 1.1160 -10012.8
## + btc_median_confirmation_time     1   0.01795 1.1376  -9985.9
## + btc_output_volume                1   0.00503 1.1505  -9970.0
## + btc_trade_volume                 1   0.00284 1.1527  -9967.4
## <none>                                         1.1556  -9965.9
## + btc_cost_per_transaction_percent 1   0.00043 1.1551  -9964.4
## + btc_transaction_fees             1   0.00039 1.1552  -9964.4
## + btc_n_unique_addresses           1   0.00038 1.1552  -9964.4
## + btc_n_transactions               1   0.00032 1.1552  -9964.3
## + btc_n_transactions_per_block     1   0.00014 1.1554  -9964.1
## - btc_hash_rate                    1   0.85930 2.0149  -9187.3
## - btc_miners_revenue               1   1.04868 2.2042  -9061.2
##
## Step:  AIC=-10018.94
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc
##
##                                   Df Sum of Sq    RSS      AIC
## + btc_avg_block_size               1   0.04026 1.0709 -10068.8
## + btc_median_confirmation_time     1   0.01883 1.0923 -10040.9
## + btc_output_volume                1   0.00339 1.1078 -10021.2
## + btc_trade_volume                 1   0.00261 1.1085 -10020.2
## <none>                                         1.1111 -10018.9
## + btc_cost_per_transaction_percent 1   0.00108 1.1101 -10018.3
## + btc_n_unique_addresses           1   0.00033 1.1108 -10017.4
## + btc_transaction_fees             1   0.00030 1.1108 -10017.3
## + btc_n_transactions               1   0.00016 1.1110 -10017.1
## + btc_n_transactions_per_block     1   0.00004 1.1111 -10017.0
## - n_btc                            1   0.04442 1.1556  -9965.9
## - btc_hash_rate                    1   0.88002 1.9911  -9201.9
## - btc_miners_revenue               1   1.07125 2.1824  -9073.2
##
## Step:  AIC=-10068.75
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size
##
##                                   Df Sum of Sq     RSS      AIC
## + btc_n_transactions               1   0.08166 0.98922 -10178.1
## + btc_n_unique_addresses           1   0.01900 1.05188 -10091.9
## + btc_output_volume                1   0.01870 1.05218 -10091.5
## + btc_transaction_fees             1   0.01550 1.05538 -10087.2
## + btc_trade_volume                 1   0.01108 1.05980 -10081.4
## + btc_median_confirmation_time     1   0.00909 1.06179 -10078.7
## + btc_cost_per_transaction_percent 1   0.00422 1.06666 -10072.3
## <none>                                         1.07088 -10068.8
## + btc_n_transactions_per_block     1   0.00000 1.07088 -10066.8
## - btc_avg_block_size               1   0.04026 1.11114 -10018.9
## - n_btc                            1   0.04511 1.11599 -10012.8
```

```
## - btc_hash_rate                      1   0.74535 1.81623  -9329.0
## - btc_miners_revenue                 1   1.09273 2.16361  -9083.3
##
## Step:  AIC=-10178.11
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_n_unique_addresses            1   0.01425 0.97497 -10196.5
## + btc_median_confirmation_time      1   0.01229 0.97693 -10193.7
## + btc_transaction_fees              1   0.00597 0.98325 -10184.6
## + btc_trade_volume                  1   0.00441 0.98481 -10182.4
## + btc_output_volume                 1   0.00188 0.98734 -10178.8
## <none>                                          0.98922 -10178.1
## + btc_cost_per_transaction_percent  1   0.00111 0.98811 -10177.7
## + btc_n_transactions_per_block      1   0.00089 0.98833 -10177.4
## - n_btc                             1   0.04148 1.03071 -10122.4
## - btc_n_transactions                1   0.08166 1.07088 -10068.8
## - btc_avg_block_size                1   0.12175 1.11098 -10017.1
## - btc_hash_rate                     1   0.60501 1.59423  -9510.1
## - btc_miners_revenue                1   1.17436 2.16359  -9081.3
##
## Step:  AIC=-10196.49
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_median_confirmation_time      1   0.01303 0.96195 -10213.4
## + btc_transaction_fees              1   0.00574 0.96923 -10202.8
## + btc_trade_volume                  1   0.00456 0.97041 -10201.1
## + btc_output_volume                 1   0.00148 0.97349 -10196.6
## <none>                                          0.97497 -10196.5
## + btc_n_transactions_per_block      1   0.00096 0.97402 -10195.9
## + btc_cost_per_transaction_percent  1   0.00090 0.97407 -10195.8
## - btc_n_unique_addresses            1   0.01425 0.98922 -10178.1
## - n_btc                             1   0.04151 1.01648 -10139.9
## - btc_n_transactions                1   0.07691 1.05188 -10091.9
## - btc_avg_block_size                1   0.13580 1.11078 -10015.4
## - btc_hash_rate                     1   0.58816 1.56313  -9535.7
## - btc_miners_revenue                1   1.18861 2.16358  -9079.3
##
## Step:  AIC=-10213.37
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s +
##     btc_median_confirmation_time
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_transaction_fees              1   0.00713 0.95481 -10221.8
## + btc_trade_volume                  1   0.00447 0.95747 -10217.9
## <none>                                          0.96195 -10213.4
## + btc_cost_per_transaction_percent  1   0.00105 0.96090 -10212.9
## + btc_output_volume                 1   0.00101 0.96093 -10212.8
```

```
## + btc_n_transactions_per_block      1   0.00018 0.96176 -10211.6
## - btc_median_confirmation_time      1   0.01303 0.97497 -10196.5
## - btc_n_unique_addresses            1   0.01499 0.97693 -10193.7
## - n_btc                             1   0.04208 1.00402 -10155.3
## - btc_n_transactions                1   0.08001 1.04195 -10103.2
## - btc_avg_block_size                1   0.12812 1.09006 -10039.8
## - btc_hash_rate                     1   0.54261 1.50455  -9587.4
## - btc_miners_revenue                1   1.20078 2.16273  -9077.9
##
## Step:  AIC=-10221.82
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s +
##     btc_median_confirmation_time + btc_transaction_fees
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_trade_volume                  1   0.00431 0.95050 -10226.2
## <none>                                          0.95481 -10221.8
## + btc_cost_per_transaction_percent  1   0.00123 0.95358 -10221.6
## + btc_output_volume                 1   0.00116 0.95366 -10221.5
## + btc_n_transactions_per_block      1   0.00028 0.95453 -10220.2
## - btc_transaction_fees              1   0.00713 0.96195 -10213.4
## - btc_median_confirmation_time      1   0.01442 0.96923 -10202.8
## - btc_n_unique_addresses            1   0.01476 0.96958 -10202.3
## - n_btc                             1   0.04199 0.99681 -10163.4
## - btc_n_transactions                1   0.07029 1.02510 -10124.1
## - btc_avg_block_size                1   0.13465 1.08946 -10038.6
## - btc_hash_rate                     1   0.53827 1.49308  -9596.1
## - btc_miners_revenue                1   1.20708 2.16190  -9076.4
##
## Step:  AIC=-10226.17
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s +
##     btc_median_confirmation_time + btc_transaction_fees + btc_trade
_volume
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_cost_per_transaction_percent  1   0.00340 0.94710 -10229.2
## <none>                                          0.95050 -10226.2
## + btc_output_volume                 1   0.00050 0.95000 -10224.9
## + btc_n_transactions_per_block      1   0.00026 0.95024 -10224.6
## - btc_trade_volume                  1   0.00431 0.95481 -10221.8
## - btc_transaction_fees              1   0.00697 0.95747 -10217.9
## - btc_median_confirmation_time      1   0.01431 0.96481 -10207.2
## - btc_n_unique_addresses            1   0.01491 0.96542 -10206.3
## - n_btc                             1   0.04187 0.99237 -10167.7
## - btc_n_transactions                1   0.06446 1.01496 -10136.0
## - btc_avg_block_size                1   0.13660 1.08710 -10039.6
## - btc_hash_rate                     1   0.53910 1.48960  -9597.4
## - btc_miners_revenue                1   1.20613 2.15663  -9077.9
##
## Step:  AIC=-10229.21
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
```

```
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s +
##     btc_median_confirmation_time + btc_transaction_fees + btc_trade
_volume +
##     btc_cost_per_transaction_percent
##
##                                    Df Sum of Sq     RSS      AIC
## + btc_output_volume                 1   0.00334 0.94377 -10232.2
## <none>                                          0.94710 -10229.2
## + btc_n_transactions_per_block      1   0.00032 0.94679 -10227.7
## - btc_cost_per_transaction_percent  1   0.00340 0.95050 -10226.2
## - btc_trade_volume                  1   0.00648 0.95358 -10221.6
## - btc_transaction_fees              1   0.00724 0.95435 -10220.5
## - btc_n_unique_addresses            1   0.01453 0.96163 -10209.8
## - btc_median_confirmation_time      1   0.01460 0.96171 -10209.7
## - n_btc                             1   0.04304 0.99014 -10168.8
## - btc_n_transactions                1   0.06737 1.01448 -10134.7
## - btc_avg_block_size                1   0.13523 1.08233 -10043.8
## - btc_hash_rate                     1   0.53880 1.48590  -9598.9
## - btc_miners_revenue                1   1.19178 2.13888  -9087.5
##
## Step:  AIC=-10232.16
## btc_market_price ~ btc_miners_revenue + btc_hash_rate + n_btc +
##     btc_avg_block_size + btc_n_transactions + btc_n_unique_addresse
s +
##     btc_median_confirmation_time + btc_transaction_fees + btc_trade
_volume +
##     btc_cost_per_transaction_percent + btc_output_volume
##
##                                    Df Sum of Sq     RSS      AIC
## <none>                                          0.94377 -10232.2
## + btc_n_transactions_per_block      1   0.00025 0.94352 -10230.5
## - btc_output_volume                 1   0.00334 0.94710 -10229.2
## - btc_trade_volume                  1   0.00619 0.94996 -10225.0
## - btc_cost_per_transaction_percent  1   0.00624 0.95000 -10224.9
## - btc_transaction_fees              1   0.00772 0.95148 -10222.7
## - btc_n_unique_addresses            1   0.01356 0.95732 -10214.1
## - btc_median_confirmation_time      1   0.01384 0.95761 -10213.7
## - n_btc                             1   0.04218 0.98594 -10172.8
## - btc_n_transactions                1   0.05944 1.00320 -10148.4
## - btc_avg_block_size                1   0.13111 1.07488 -10051.5
## - btc_hash_rate                     1   0.54156 1.48533  -9597.4
## - btc_miners_revenue                1   1.19498 2.13875  -9085.6
```

```
summary(model.step)
```

```
##
## Call:
## lm(formula = btc_market_price ~ btc_miners_revenue + btc_hash_rate
+
##     n_btc + btc_avg_block_size + btc_n_transactions + btc_n_unique_
addresses +
##     btc_median_confirmation_time + btc_transaction_fees + btc_trade
_volume +
```

```
##     btc_cost_per_transaction_percent + btc_output_volume, data = bt
c_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.128051 -0.012239 -0.002667  0.008509  0.277497
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>
|t|)
## (Intercept)                     0.0042290  0.0007027   6.018 2.25
e-09
## btc_miners_revenue              0.5749640  0.0136953  41.983  < 2
e-16
## btc_hash_rate                  -0.3979172  0.0140793 -28.263  < 2
e-16
## n_btc                          -0.8653454  0.1097150  -7.887 6.20
e-15
## btc_avg_block_size              0.1377217  0.0099037  13.906  < 2
e-16
## btc_n_transactions             -0.1010156  0.0107886  -9.363  < 2
e-16
## btc_n_unique_addresses         -0.0277232  0.0061995  -4.472 8.39
e-06
## btc_median_confirmation_time    0.0217476  0.0048130   4.518 6.76
e-06
## btc_transaction_fees           -0.0107806  0.0031955  -3.374 0.00
0762
## btc_trade_volume               -0.0041553  0.0013747  -3.023 0.00
2551
## btc_cost_per_transaction_percent -0.0093131  0.0030708  -3.033 0.00
2468
## btc_output_volume              -0.0062014  0.0027959  -2.218 0.02
6716
##
## (Intercept)                     ***
## btc_miners_revenue              ***
## btc_hash_rate                   ***
## n_btc                           ***
## btc_avg_block_size              ***
## btc_n_transactions              ***
## btc_n_unique_addresses          ***
## btc_median_confirmation_time    ***
## btc_transaction_fees            ***
## btc_trade_volume                **
## btc_cost_per_transaction_percent **
## btc_output_volume               *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02604 on 1392 degrees of freedom
## Multiple R-squared:  0.5721, Adjusted R-squared:  0.5687
## F-statistic: 169.2 on 11 and 1392 DF,  p-value: < 2.2e-16
```

Now, this makes much more sense. The variable trans_to_trade was biasing our estimator as it contained the price itself. The only variables having a possitive effect on price are the miners revenue, the average block size in MB and the median confirmation time. That means that price is increased when there is more activity in the network, so that blocks store more information and the network is slightly saturated, which causes the confirmation time to be higher; when miners receive more revenues it means that the price is increasing, since they get more dollar revenue per their activity when bitcoins are worth more dollars. The variables hash rate, number of btc, number of transactions, number of unique addresses, transaction fees, trade volume, cost per transaction percent and output volume negatively affect the price. The R^2 value shows that there may be some explanatory variables missing, like investors sentiment, news, and others. This shows which variables could be potentially used, in shorter time intervals and retrieving data directly from the Blockchain on real-time, to predict the future bitcoin prices. However, I will not go deep in the analysis of each of them, as it is not the objective of this project to describe or explain bitcoin price but to predict it in the future.

```
dif2 <- diff_btc
dif2$btc_market_price <- c(dif2$btc_market_price[2:nrow(diff_btc)], NA
) #this way, we are trying to see if the variables in t-1 can predict
future log return in t.

dif2 <- dif2[-nrow(dif2), ]

train_index <- seq(1, 0.8*nrow(dif2), 1)
#train = diff_btc[1:0.7*nrow(diff_btc), ]
btc_train <- dif2[train_index, ]
btc_test <- dif2[-train_index, ]

nullmodel = lm(btc_market_price ~1, data = btc_train)
fullmodel = lm(btc_market_price ~., data = btc_train)

model.step=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel
), direction='both')

## Start:  AIC=-9080.97
## btc_market_price ~ 1
##
##                                   Df Sum of Sq    RSS     AIC
## + btc_n_transactions_per_block     1 0.0056740 2.1709 -9082.6
## + btc_hash_rate                    1 0.0033082 2.1732 -9081.1
## <none>                                         2.1765 -9081.0
## + n_btc                            1 0.0028116 2.1737 -9080.8
## + btc_miners_revenue               1 0.0025197 2.1740 -9080.6
## + btc_avg_block_size               1 0.0013070 2.1752 -9079.8
## + btc_trade_volume                 1 0.0010262 2.1755 -9079.6
```

```
## + btc_cost_per_transaction_percent  1 0.0009626 2.1756 -9079.6
## + btc_n_transactions                1 0.0005210 2.1760 -9079.3
## + btc_output_volume                 1 0.0004329 2.1761 -9079.3
## + btc_transaction_fees              1 0.0001842 2.1763 -9079.1
## + btc_median_confirmation_time      1 0.0001557 2.1764 -9079.1
## + btc_n_unique_addresses            1 0.0000058 2.1765 -9079.0
##
## Step:  AIC=-9082.64
## btc_market_price ~ btc_n_transactions_per_block
##
##                                    Df Sum of Sq    RSS     AIC
## <none>                                          2.1709 -9082.6
## + n_btc                             1 0.0030509 2.1678 -9082.6
## + btc_hash_rate                     1 0.0022329 2.1686 -9082.1
## + btc_miners_revenue                1 0.0016969 2.1692 -9081.7
## + btc_avg_block_size                1 0.0010508 2.1698 -9081.3
## + btc_trade_volume                  1 0.0010493 2.1698 -9081.3
## + btc_cost_per_transaction_percent  1 0.0008302 2.1700 -9081.2
## + btc_median_confirmation_time      1 0.0007889 2.1701 -9081.1
## - btc_n_transactions_per_block      1 0.0056740 2.1765 -9081.0
## + btc_output_volume                 1 0.0005103 2.1703 -9081.0
## + btc_n_transactions                1 0.0004589 2.1704 -9080.9
## + btc_transaction_fees              1 0.0001230 2.1707 -9080.7
## + btc_n_unique_addresses            1 0.0000116 2.1708 -9080.6
```

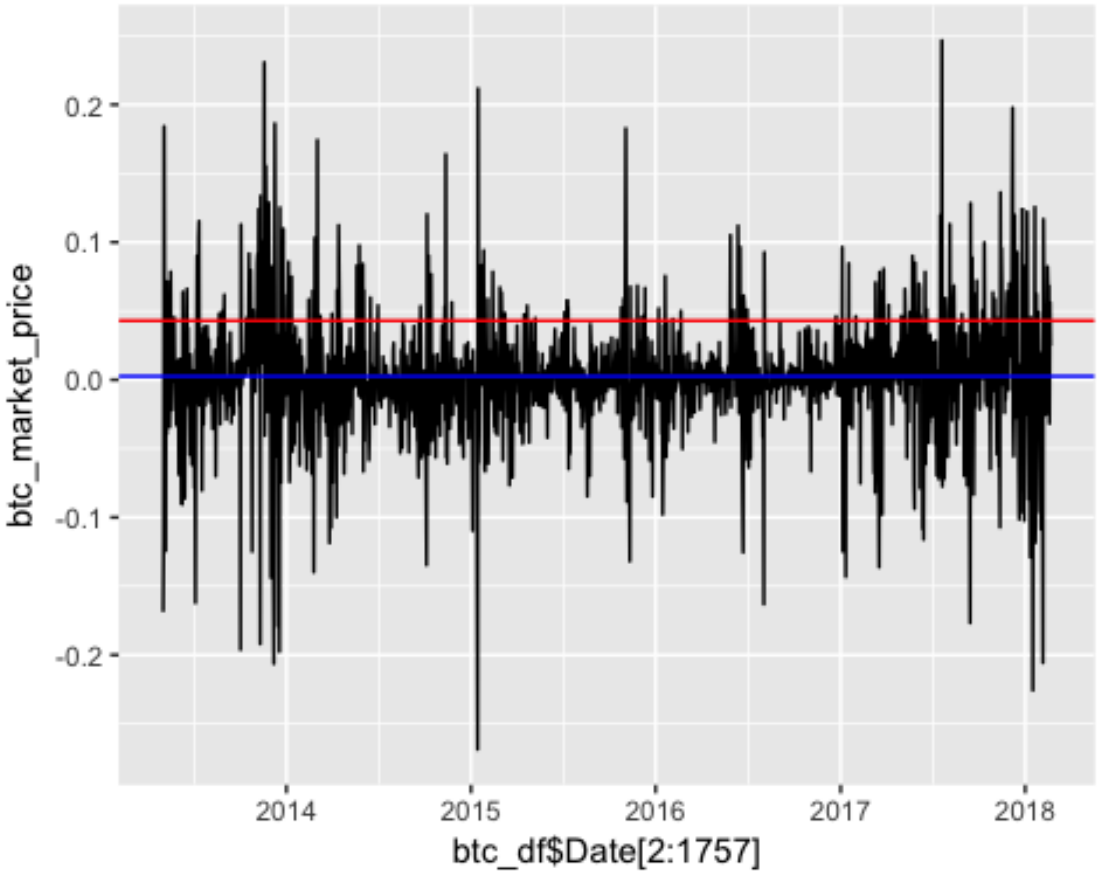
```
summary(model.step)
```

```
##
## Call:
## lm(formula = btc_market_price ~ btc_n_transactions_per_block,
##     data = btc_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.271911 -0.012446 -0.000122  0.012841  0.228440
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.001795   0.001050   1.709   0.0876
.
## btc_n_transactions_per_block -0.013031   0.006807  -1.914   0.0558
.
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03935 on 1402 degrees of freedom
## Multiple R-squared:  0.002607,   Adjusted R-squared:  0.001896
## F-statistic: 3.664 on 1 and 1402 DF,  p-value: 0.05579
```

It is very clear that the explanatory variables are only useful at time t, as we cannot predict the diff log price at time t with the rest of variables at time t-1.

```
library(ggplot2)
```

```
ggplot(data = diff_btc, aes(x = btc_df$Date[2:1757])) +
  geom_line(aes(y = btc_market_price)) +
  geom_hline(yintercept = sd(diff_btc$btc_market_price), color = 'red'
) +
  geom_hline(yintercept = mean(diff_btc$btc_market_price), color = 'bl
ue')
```



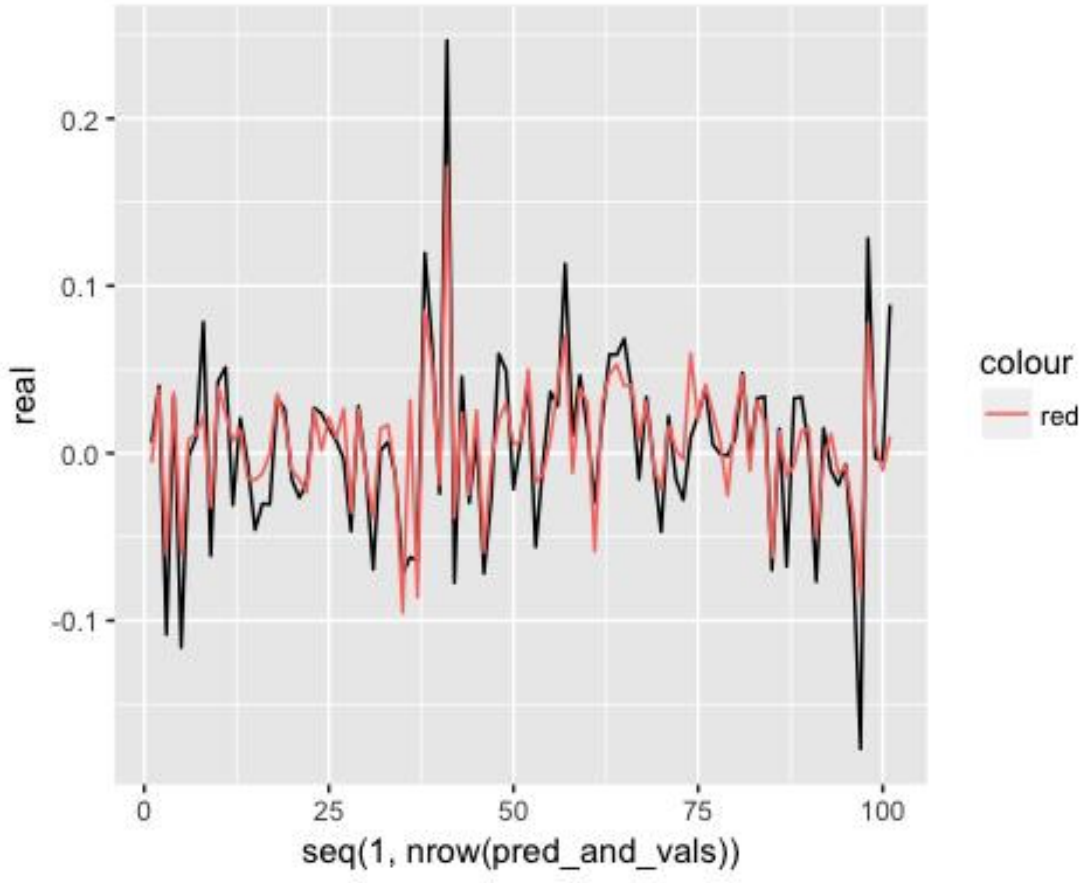

```
#now let's try to fit an arima model to this.
library(forecast)
xreg = data.frame(cbind(diff_btc$btc_hash_rate, diff_btc$btc_miners_re
venue, diff_btc$n_btc, diff_btc$btc_transaction_fees, diff_btc$btc_med
ian_confirmation_time, diff_btc$btc_avg_block_size, diff_btc$btc_n_uni
que_addresses, diff_btc$btc_cost_per_transaction_percent, diff_btc$btc
_output_volume, diff_btc$btc_n_transactions))

arimax_model <- auto.arima(diff_btc$btc_market_price[1:1500], xreg = x
reg[1:1500, ])
summary(arimax_model) #it seems that the Hessian Matrix method used gi
ves NaN and 0's in the s.e. of some variables; therefore this model is
not useful as it also does not provide a MPE or MAPE for comparing for
ecasts with other methods.

## Series: diff_btc$btc_market_price[1:1500]
## Regression with ARIMA(4,1,3) errors
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ma1     ma2      ma3
X1
```

```
##       1.5513  -0.8433  0.0321  -0.0833  -2.5978  2.4707  -0.8666  -
0.4452
## s.e.  0.0395   0.0541  0.0509   0.0314   0.0307  0.0491   0.0247
0.0137
##           X2       X3       X4      X5      X6       X7       X8
X9
##       0.6059  -0.8474  -0.0125  0.0160  0.1277  -0.0244  -0.0057  -
0.0047
## s.e.  0.0133   0.1025   0.0032  0.0041  0.0095   0.0060   0.0028
0.0026
##           X10
##       -0.0932
## s.e.   0.0102
##
## sigma^2 estimated as 0.0006473:  log likelihood=3382.96
## AIC=-6729.91   AICc=-6729.45   BIC=-6634.29
##
## Training set error measures:
##                        ME       RMSE        MAE MPE MAPE      MASE
## Training set 0.0002895638 0.02528807 0.01595795 NaN  Inf 0.4558184
##                      ACF1
## Training set 0.0003999918
```

```
pred <- forecast(object = arimax_model, xreg = xreg[1501:1601, ], h =
100 )
summary(pred)
```

```
##
## Forecast method: Regression with ARIMA(4,1,3) errors
##
## Model Information:
## Series: diff_btc$btc_market_price[1:1500]
## Regression with ARIMA(4,1,3) errors
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ma1     ma2      ma3
X1
##       1.5513  -0.8433  0.0321  -0.0833  -2.5978  2.4707  -0.8666  -
0.4452
## s.e.  0.0395   0.0541  0.0509   0.0314   0.0307  0.0491   0.0247
0.0137
##           X2       X3       X4      X5      X6       X7       X8
X9
##       0.6059  -0.8474  -0.0125  0.0160  0.1277  -0.0244  -0.0057  -
0.0047
## s.e.  0.0133   0.1025   0.0032  0.0041  0.0095   0.0060   0.0028
0.0026
##           X10
##       -0.0932
## s.e.   0.0102
##
## sigma^2 estimated as 0.0006473:  log likelihood=3382.96
## AIC=-6729.91   AICc=-6729.45   BIC=-6634.29
##
```

```
## Error measures:
##                         ME       RMSE        MAE MPE MAPE      MASE
## Training set 0.0002895638 0.02528807 0.01595795 NaN  Inf 0.4558184
##                      ACF1
## Training set 0.0003999918
##
## Forecasts:
##      Point Forecast        Lo 80         Hi 80        Lo 95
Hi 95
## 1501  -0.0057691762 -0.038373357  0.0268350047 -0.055632968  0.0440
94616
#etc

pred_and_vals <- cbind(diff_btc[1501:1601, 'btc_market_price'], pred$m
ean)
colnames(pred_and_vals) <- c('real', 'pred')
pred_and_vals <- data.frame(pred_and_vals)
library(ggplot2)
ggplot(data = pred_and_vals , aes(x = seq(1, nrow(pred_and_vals)))) +
  geom_line(aes(y = real)) +
  geom_line(aes(y = pred, color = "red")) #the prediction seems to fit
very good; this stresses the importance of the explanatory variables f
ound; however, for the aforementioned causes, this is not useful for a
ctual forecasting one day ahead, at least with daily data. Nevertheles
s, it provides some insight into the nature of the btc price.
```




```
performance <- accuracy(f = pred_and_vals$pred, x = pred_and_vals$real
)
```

```r
#install.packages("ModelMetrics")
library(ModelMetrics)
mse_arimax <- mse(actual = pred_and_vals$real, predicted = pred_and_va
ls$pred)


accu = 1-performance[5]

#print(paste("The accuracy of such a model measured in terms of MAPE i
s:", accu))
print(paste("The accuracy of such a model measured in terms of MSE is:
", mse_arimax))

## [1] "The accuracy of such a model measured in terms of MSE is: 0.00
0766704828515717"
```

This brief summary therefore contains an exploration for the discovery of some possible variables that can be explanatory of bitcoin log return. However, these are not useful for forecasting for the aforementioned causes. Therefore, the focus will on looking for predictive methods. Nevertheless, this exploration brings the conclusion that taking sufficient information via web scraping from the Blockchain itself (the variables used can all be extracted on almost real time from the Blockchain), the future price of bitcoin could be very well estimated. For this, shorter time intervals should be used than the ones used here.

Before going directly into ARIMA, the authors mentioned above claimed to have more than 98% of accuracy (in this case measured in terms of correct rate) using a Binomial GLM with these characteristics. As seen above, a linear regression using these variables for forecasting one day ahead does not provide any reliable variable, as none of them is significant at 5% alpha. I will try to replicate their results, to see if these variables could be useful in classifying the next price as going up (1) or going down (0), using the default logit function from the binomial family of Generalized Linear Models.

```r
dif3 <- apply(btc_df[ ,2:ncol(btc_df)], MARGIN = 2, FUN = diflog)
#the variable difficulty is almost always 0; therefore we will take it
out.
dif3 <- dif3[, -11]
dif3 <- data.frame(dif3)

dif3$next_price <- c(dif3$btc_market_price[2:nrow(dif3)], NA) #this wa
y, we are trying to see if the variables in t-1 can predict future log
return in t.

dif3 <- dif3[-nrow(dif3), ]
#now we will try to replicate the work by Majan, Saluja and Zhao using
```

```r
Binomial GLM.
dif3$change <- c()

for (i in 1:nrow(dif3)) {
  if(dif3$next_price[i] > 0) {
    dif3$change[i] <- 1
  } else #if (dif3$next_price < 0)
    {
    dif3$change[i] <- 0
  }
}

#dif3 <- dif3[, -1] #by the moment we keep the previous day's market p
rice change (diff); maybe it is a good predictor of next day's price c
hange.
dif3 <- dif3[, -15]

train_index <- seq(1, 0.8*nrow(dif3), 1)
btc_train <- dif3[train_index, ]
btc_test <- dif3[-train_index, ]

nullmodel <- glm(change ~1 , data = btc_train, family = 'binomial')
fullmodel <- glm(change ~., data = btc_train, family = 'binomial')

model.step=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel
), direction='both')

## Start:  AIC=1938.09
## change ~ 1
##
##                                    Df Deviance    AIC
## + btc_trade_volume                  1   1933.3 1937.3
## + btc_avg_block_size                1   1933.7 1937.7
## + btc_output_volume                 1   1933.7 1937.7
## + btc_hash_rate                     1   1933.9 1937.9
## <none>                                  1936.1 1938.1
## + btc_n_transactions_per_block      1   1934.8 1938.8
## + btc_n_unique_addresses            1   1935.1 1939.1
## + btc_transaction_fees              1   1935.1 1939.1
## + btc_miners_revenue                1   1935.2 1939.2
## + btc_cost_per_transaction_percent  1   1935.6 1939.6
## + btc_market_price                  1   1935.8 1939.8
## + trans_to_trade                    1   1935.8 1939.8
## + btc_n_transactions                1   1936.0 1940.0
## + btc_median_confirmation_time      1   1936.1 1940.1
## + n_btc                             1   1936.1 1940.1
##
## Step:  AIC=1937.29
## change ~ btc_trade_volume
##
##                                    Df Deviance    AIC
## + btc_avg_block_size                1   1929.2 1935.2
## + btc_hash_rate                     1   1930.8 1936.8
## <none>                                  1933.3 1937.3
```

```
## + btc_transaction_fees               1   1931.7 1937.7
## + btc_n_unique_addresses             1   1931.8 1937.8
## + btc_n_transactions_per_block       1   1932.0 1938.0
## + btc_output_volume                  1   1932.0 1938.0
## - btc_trade_volume                   1   1936.1 1938.1
## + btc_miners_revenue                 1   1932.3 1938.3
## + trans_to_trade                     1   1933.0 1939.0
## + btc_market_price                   1   1933.0 1939.0
## + btc_n_transactions                 1   1933.2 1939.2
## + btc_median_confirmation_time       1   1933.3 1939.3
## + btc_cost_per_transaction_percent   1   1933.3 1939.3
## + n_btc                              1   1933.3 1939.3
##
## Step:  AIC=1935.25
## change ~ btc_trade_volume + btc_avg_block_size
##
##                                    Df Deviance    AIC
## + btc_output_volume                 1   1926.5 1934.5
## <none>                                  1929.2 1935.2
## + btc_n_transactions                1   1928.0 1936.0
## + btc_cost_per_transaction_percent  1   1928.1 1936.1
## + btc_n_transactions_per_block      1   1928.2 1936.2
## + btc_median_confirmation_time      1   1928.3 1936.3
## + btc_transaction_fees              1   1928.8 1936.8
## + btc_hash_rate                     1   1929.0 1937.0
## + btc_n_unique_addresses            1   1929.0 1937.0
## + btc_market_price                  1   1929.1 1937.1
## + trans_to_trade                    1   1929.1 1937.1
## + btc_miners_revenue                1   1929.2 1937.2
## + n_btc                             1   1929.2 1937.2
## - btc_avg_block_size                1   1933.3 1937.3
## - btc_trade_volume                  1   1933.7 1937.7
##
## Step:  AIC=1934.55
## change ~ btc_trade_volume + btc_avg_block_size + btc_output_volume
##
##                                    Df Deviance    AIC
## <none>                                  1926.5 1934.5
## + btc_median_confirmation_time      1   1925.2 1935.2
## - btc_output_volume                 1   1929.2 1935.2
## - btc_trade_volume                  1   1929.3 1935.3
## + btc_n_transactions_per_block      1   1925.6 1935.6
## + btc_transaction_fees              1   1925.9 1935.9
## + btc_n_unique_addresses            1   1926.0 1936.0
## + btc_n_transactions                1   1926.3 1936.3
## + btc_miners_revenue                1   1926.4 1936.4
## + trans_to_trade                    1   1926.5 1936.5
## + btc_market_price                  1   1926.5 1936.5
## + btc_cost_per_transaction_percent  1   1926.5 1936.5
## + btc_hash_rate                     1   1926.5 1936.5
## + n_btc                             1   1926.5 1936.5
## - btc_avg_block_size                1   1932.0 1938.0

summary(model.step)
```

```
##
## Call:
## glm(formula = change ~ btc_trade_volume + btc_avg_block_size +
##     btc_output_volume, family = "binomial", data = btc_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.508  -1.235   1.003   1.107   1.399
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.17371    0.05377   3.231  0.00123 **
## btc_trade_volume     0.17146    0.10285   1.667  0.09549 .
## btc_avg_block_size -0.90074    0.38620  -2.332  0.01968 *
## btc_output_volume    0.28890    0.17689   1.633  0.10243
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1936.1  on 1403  degrees of freedom
## Residual deviance: 1926.6  on 1400  degrees of freedom
## AIC: 1934.6
##
## Number of Fisher Scoring iterations: 4
```

```
pred <- predict(object = model.step, newdata = btc_test, "response") #
preguntar a ricardo por que coño saca esto valores numericos y no 0 o
1, que es lo que en teoria nos deberia devolver.
pred
```

```
##      1405      1406      1407      1408      1409      1410      14
11
## 0.5930144 0.5886929 0.5549587 0.5795019 0.5444904 0.4615746 0.50987
74
##      1412      1413      1414      1415      1416      1417      14
18
## 0.5882899 0.5546776 0.5234522 0.6085464 0.5437240 0.5483189 0.59550
88
etc
```

```
pred2 <- ifelse(pred>0.5, 1, 0)

btc_test$pred <- pred2
btc_test$correct <- c()

for(i in 1:nrow(btc_test)) {
  if(btc_test$pred[i] == btc_test$change[i]) {
    btc_test$correct[i] <- 1
  } else {
    btc_test$correct[i] <- 0
  }
}

percentage_correct <- sum(btc_test$correct) / nrow(btc_test)
```

```r
print(percentage_correct) #only 59% of correct assessment of price goi
ng up or down; therefore this model does not replicate the one by the
above authors. Actually this accuracy is almost like tossing a coin, a
s it is only a little bit above 50%, which would be the randomly expec
ted generated prediction accuracy.
```

```
## [1] 0.5925926
```

The conclusion is therefore that the results attained by Madan, Saluja and Zhao (2014) can be because they use data from the same day to classify the closing price of that same day (they do not specify too much the method followed, so this is not testeable). Other possibility is that their results hold up to 2014, but these do not hold anymore. For this reason, the next step will be to use an ARIMA model to forecast bitcoin and ether prices, without taking these external variables into account, as the objective is to be able to design a trading strategy, and the ARIMAX used above uses data from the same day as external regressors (which, on the other hand, is something required by the model architecture: after fitting the model with the external regressors, for forecasting, the external regressors have to be included, which correspond to time t instead of time t-1).

ANNEX 3

```r
library(readr)

btc <- read_csv('btc_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

eth <- read_csv('eth_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

btc <- btc[ , 2:ncol(btc)]
eth <- eth[ , 2:ncol(eth)]

colnames(btc) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")
colnames(eth) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
btc$date <- as.Date(btc$date)
eth$date <- as.Date(eth$date)
btc1 <- btc %>%
  filter(date > "2013-05-01" & date < "2017-10-01")

eth1 <- eth %>%
  filter(date > "2016-05-01" & date < "2017-10-01") #for ethereum the forecast length will be smaller, as there is less data for this crypto currency. Therefore, the forecast will be from day: "2017-06-05" to "2017-09-30"

pred1_btc <- c()
library(forecast)

for(i in 900:nrow(btc1)) {  #this forecast correspond from day: "2015-10-18" to "2017-09-30"
  arima <- auto.arima(btc1$close[1:i-1])
  forec <- forecast(object = arima, h = 1)
  pred1_btc <- c(pred1_btc, forec$mean)
}

acc <- accuracy(f = pred1_btc, x = btc1$close[900:nrow(btc1)])
print(acc)
```

```
##                 ME     RMSE     MAE       MPE     MAPE
## Test set 3.547527 81.91822 37.4481 0.2167208 2.475729
```

```r
accuracy_btc <- 100 - acc[5] #we define the accuracy in % as 100 - MAPE (which is located in acc[5])
print(paste("the accuracy of btc in this first time frame is" , accuracy_btc, "%"))
```

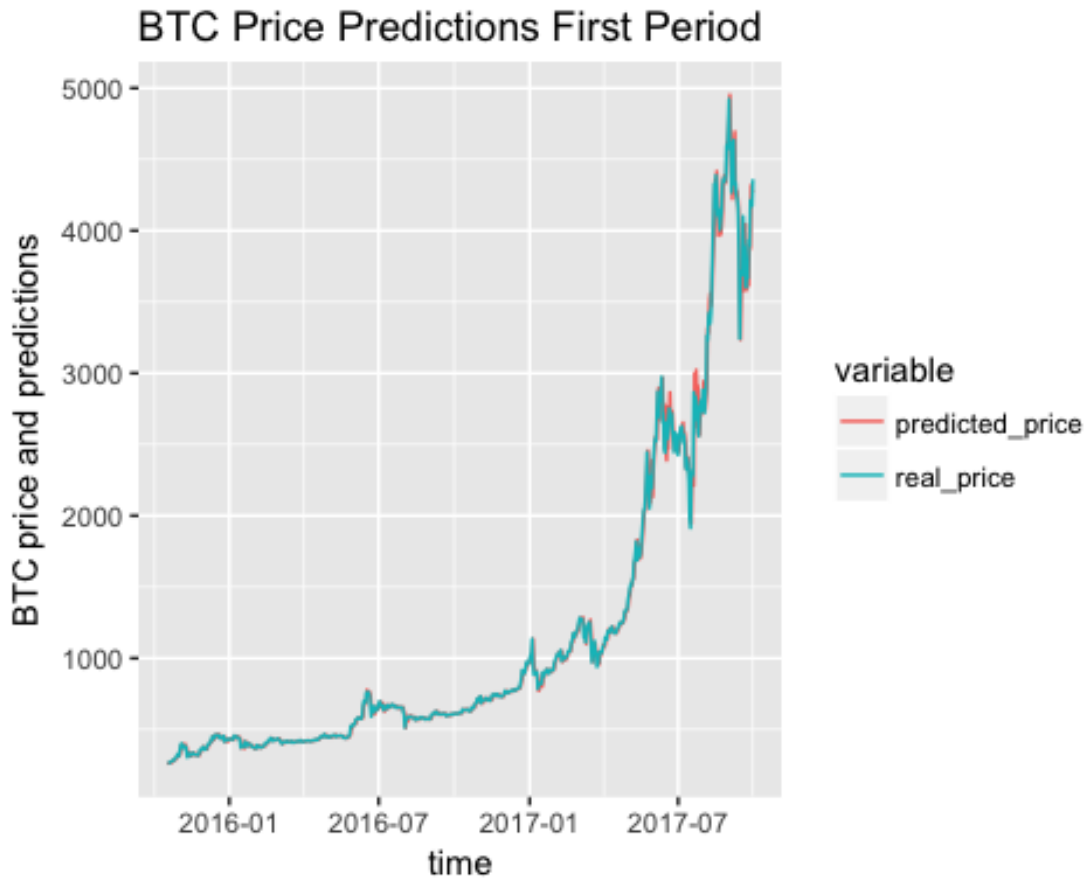
```
## [1] "the accuracy of btc in this first time frame is 97.52427075981 25 %"
```

```r
library(ModelMetrics)
mse_btc1 <- mse(actual = btc1$close[900:nrow(btc1)], predicted = pred1_btc)
print(mse_btc1) #although the mse is displayed, it will not be used for comparing predictions as it does not make sense to compare two different assets with different magnitudes (one in hundreds and the other in thousands) by the error value; it is better to use a measure relative to the magnitude of each.
```

```
## [1] 6710.596
```

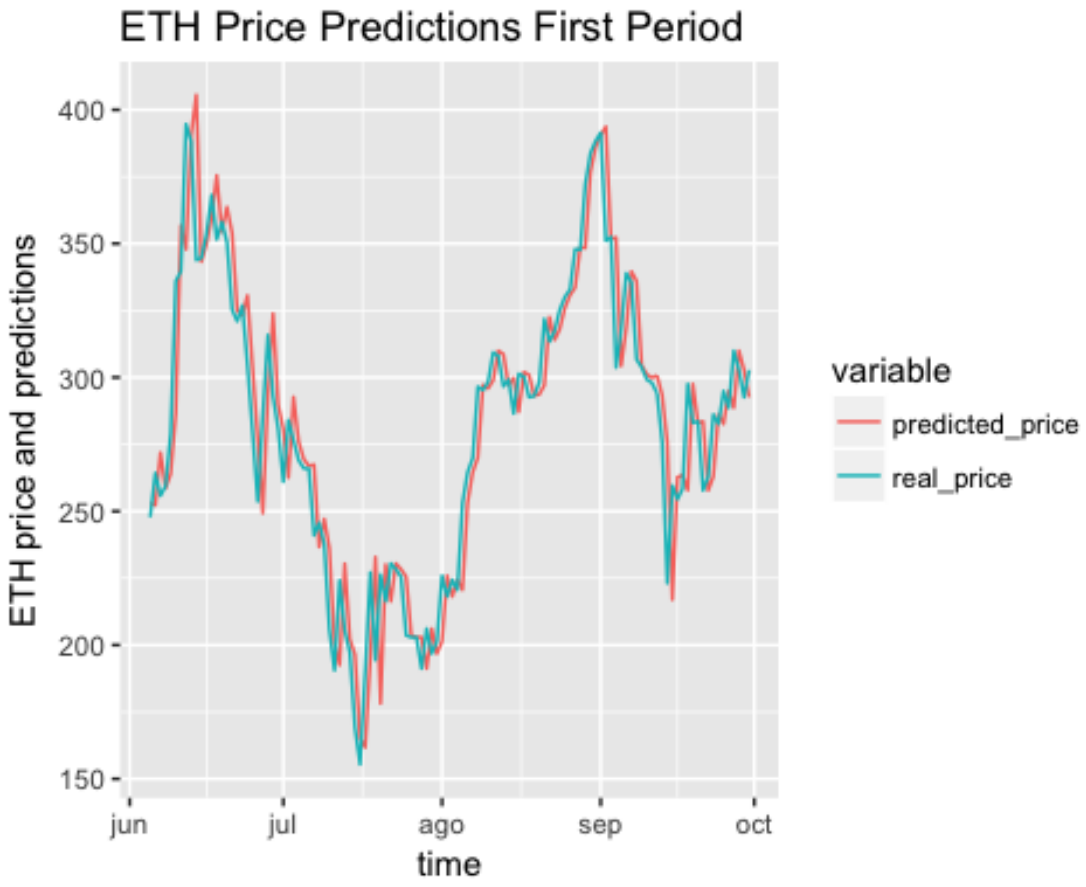
```r
close_prices <- btc1$close[900:nrow(btc1)]
x <- c(btc1$date[900:nrow(btc1)], btc1$date[900:nrow(btc1)])
variable <- c(rep("real_price", 714), rep("predicted_price", 714))
value <- c(close_prices, pred1_btc)
btc_pred <- data.frame(x, variable, value)

library(ggplot2)

ggplot(btc_pred, aes(x = x, y = value, group = variable, colour = vari
```

```
able)) +
  geom_line() +
  xlab("time") +
  ylab("BTC price and predictions") +
  ggtitle("BTC Price Predictions First Period")
```




```
library(forecast)
pred1_eth <- c()

for(i in 400:nrow(eth1)) {
  arima <- auto.arima(eth1$close[1:i-1])
  forec <- forecast(object = arima, h = 1 )
  pred1_eth <- c(pred1_eth, forec$mean)
}


accuracy <- accuracy(f = pred1_eth, x = eth1$close[400:nrow(eth1)])
mse_eth1 <- mse(actual = eth1$close[400:nrow(eth1)], predicted = pred1
_eth)
print(mse_eth1)

## [1] 406.6686

#the MAPE is used, as it enables us to compare the forecast performanc
e in different cryptocurrencies.

print(paste("the MAPE for ethereum in this first attempt is", accuracy
[5]))
```

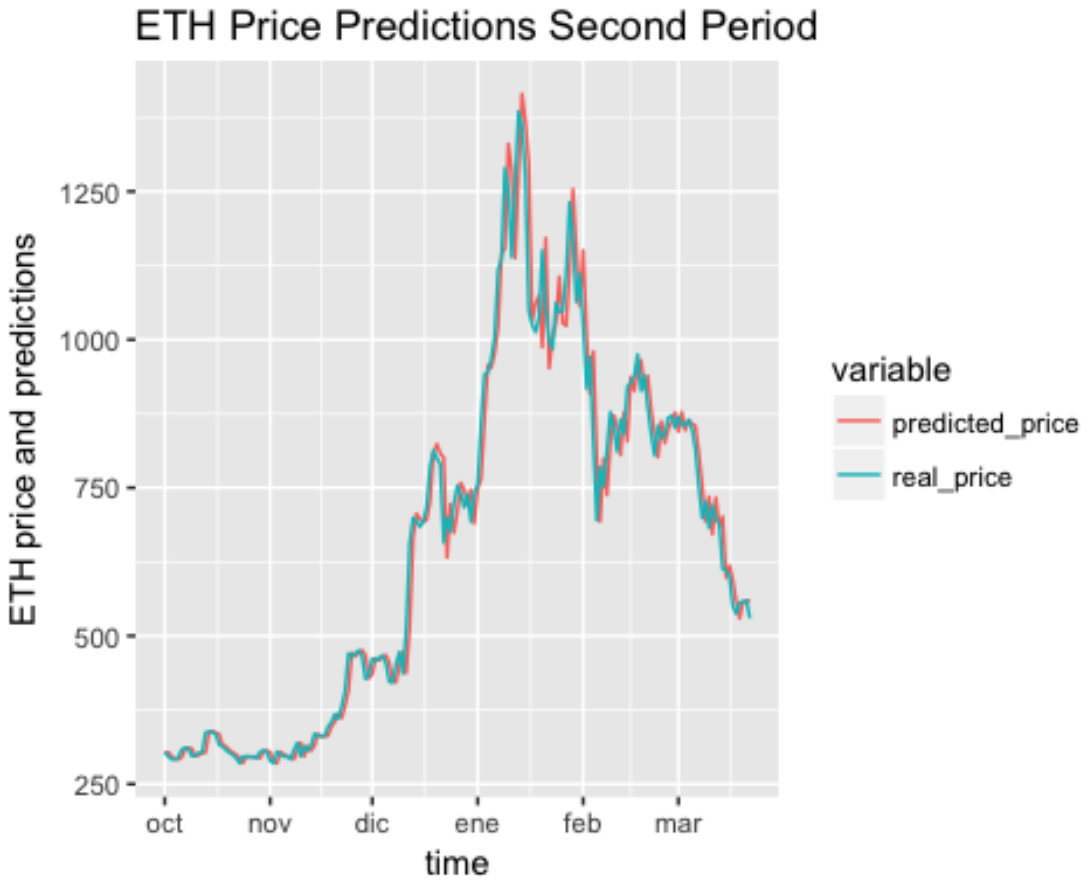
```
## [1] "the MAPE for ethereum in this first attempt is 5.5533235186398
8"

print(paste("the accuracy for ethereum in this first attempt is", 100
- accuracy[5], "%"))

## [1] "the accuracy for ethereum in this first attempt is 94.44667648
13601 %"

close_prices <- eth1$close[400:nrow(eth1)]
x <- c(eth1$date[400:nrow(eth1)], eth1$date[400:nrow(eth1)])
variable <- c(rep("real_price", 118), rep("predicted_price", 118))
value <- c(close_prices, pred1_eth)
eth_pred <- data.frame(x, variable, value)

library(ggplot2)

ggplot(eth_pred, aes(x = x, y = value, group = variable, colour = vari
able)) +
  geom_line() +
  xlab("time") +
  ylab("ETH price and predictions") +
  ggtitle("ETH Price Predictions First Period")
```




```
                SECOND TIME PERIOD (2017-10-01 / 2018-03-22)
```

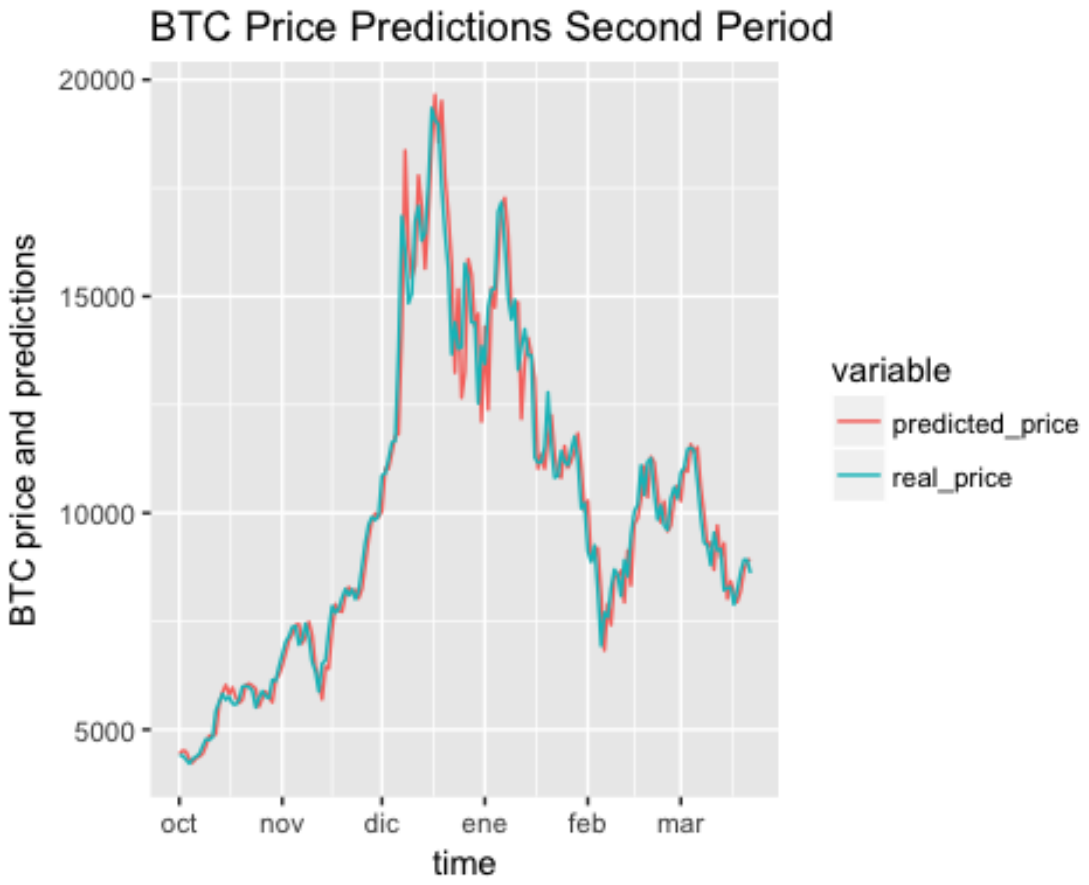
```r
library(readr)

btc <- read_csv('btc_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

eth <- read_csv('eth_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

btc <- btc[ , 2:ncol(btc)]
eth <- eth[ , 2:ncol(eth)]

colnames(btc) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")
colnames(eth) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")

library(dplyr)
btc$date <- as.Date(btc$date)
eth$date <- as.Date(eth$date)
btc1 <- btc %>%
  filter(date > "2013-05-01")

eth1 <- eth %>%
  filter(date > "2016-05-01" )

for(i in 1:nrow(eth1)) {
  if(eth1$date[i] == "2017-10-01") {
    print(i)
    break
  } else {
    next
  }
}  #the first forecast value corresponds to row 518 in case of eth
```

```
## [1] 518

for(i in 1:nrow(btc1)) {
  if(btc1$date[i] == "2017-10-01") {
    print(i)
    break
  } else {
    next
  }
}  #the first forecast value corresponds to row 1614 in case of btc

## [1] 1614

pred2_eth <- c()

for(i in 518:nrow(eth1)) {
  arima <- auto.arima(eth1$close[1:i-1])
  forec <- forecast(object = arima, h = 1 )
  pred2_eth <- c(pred2_eth, forec$mean)
}


acc <- accuracy(f = pred2_eth, x = eth1$close[518:nrow(eth1)])
print(acc)

##                  ME     RMSE      MAE         MPE    MAPE
## Test set -0.840582 53.37602 35.18303 -0.08535107 4.69951

mse_eth2 <- mse(actual = eth1$close[518:nrow(eth1)], predicted = pred2
_eth)
print(mse_eth2)

## [1] 2848.999

print(paste("the accuracy for ether in the second period is", 100 - ac
c[5], "%"))

## [1] "the accuracy for ether in the second period is 95.300489990720
1 %"

close_prices <- eth1$close[518:nrow(eth1)]
x <- c(eth1$date[518:nrow(eth1)], eth1$date[518:nrow(eth1)])
variable <- c(rep("real_price", 173), rep("predicted_price", 173))
value <- c(close_prices, pred2_eth)
eth_pred <- data.frame(x, variable, value)

library(ggplot2)

ggplot(eth_pred, aes(x = x, y = value, group = variable, colour = vari
able)) +
  geom_line() +
  xlab("time") +
  ylab("ETH price and predictions") +
  ggtitle("ETH Price Predictions Second Period")
```




```
pred2_btc <- c()
library(forecast)

for(i in 1614:nrow(btc1)) {  #this forecast correspond from day: "2017
-10-01" to "2018-03-22"
  arima <- auto.arima(btc1$close[1:i-1])
  forec <- forecast(object = arima, h = 1)
  pred2_btc <- c(pred2_btc, forec$mean)
}

accuracy(f = pred2_btc, x = btc1$close[1614:nrow(btc1)])

##                  ME     RMSE      MAE          MPE     MAPE
## Test set -2.929411 766.2197 533.3653 -0.05400942 4.928709

library(ModelMetrics)
mse_btc2 <- mse(actual = btc1$close[1614:nrow(btc1)], predicted = pred
2_btc)
print(mse_btc2)

## [1] 587092.6

close_prices <- btc1$close[1614:nrow(btc1)]
x <- c(btc1$date[1614:nrow(btc1)], btc1$date[1614:nrow(btc1)])
variable <- c(rep("real_price", 173), rep("predicted_price", 173))
value <- c(close_prices, pred2_btc)
btc_pred <- data.frame(x, variable, value)
```

```
library(ggplot2)

ggplot(btc_pred, aes(x = x, y = value, group = variable, colour = vari
able)) +
  geom_line() +
  xlab("time") +
  ylab("BTC price and predictions") +
  ggtitle("BTC Price Predictions Second Period")
```




```
                                ADDING GARCH (1,1) TO THE MODEL
```

In many cases, the name of the variables is repeated, in order to avoid creating too large names. This poses no trouble for the task, and following the logical order of the computations it should not be a problem for the reader to follow which variable represents what at each moment in time. However, for this, the order of the code must be followed as it was designed, from top to botton. First, the below code gets the optimal window for which MPE is minimized. It is not run due to the time requirements; however, this poses no problem as it was already previously run, arriving at the conclusion that 400 days is the best window.

```
require(readr)
btc = read_csv('btc_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
```

```
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

head(btc)

## # A tibble: 6 x 8
##       time timeDate    close   high    low   open volumefrom volumet
o
##      <dbl> <date>      <dbl>  <dbl>  <dbl>  <dbl>      <dbl>    <dbl
>
## 1 1.28e12 2010-07-17 0.0495 0.0495 0.0495 0.0495       20.0    0.99
0
## 2 1.28e12 2010-07-18 0.0858 0.0858 0.0594 0.0495       75.0    5.09
## 3 1.28e12 2010-07-19 0.0808 0.0931 0.0772 0.0858      574.    49.7
## 4 1.28e12 2010-07-20 0.0747 0.0818 0.0743 0.0808      262.    20.6
## 5 1.28e12 2010-07-21 0.0792 0.0792 0.0663 0.0747      575.    42.3
## 6 1.28e12 2010-07-22 0.0505 0.0818 0.0505 0.0792     2160.   130.

btc = btc[, 2:ncol(btc)]
btc$timeDate = as.Date(btc$timeDate)
class(btc$timeDate)

## [1] "Date"

require(dplyr)
btc = btc %>%
  filter(timeDate > '2013-04-15')

btc2 = btc %>%
  filter(timeDate > '2014-11-15')

prices = btc2$close

require(rugarch)

## Loading required package: rugarch

## Loading required package: parallel

##
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':
##
##     sigma

require(forecast)
require()

## Loading required package:
```

```
pred3 = c()
length(prices)

## [1] 1223

windowLength <- 400
foreLength = length(prices) - windowLength
best.accuracy <- 2
best.window <- 0

#this is the code used for arriving at the best window; it will not ru
n now as it took almost a week for it to get a result; I leave it here
in case someone wants to replicate my work with their own computer:



for(i in 400:600) {
  windowLength = i
  foreLength = length(prices) - windowLength
  real_pr <- prices[windowLength:length(prices)]

  for (d in 0:foreLength) {
    # Obtain the bitcoin rolling window for this day
    pricesOffset = prices[(1+d):(windowLength+d)]

    # Fit the ARIMA model
    final.aic <- Inf
    final.order <- c(0,0,0)
    for (p in 0:5) for (q in 0:5) {  #with this the objective is to fi
nd the best p,q orders in arima (AR and MA components)
      if ( p == 0 && q == 0) {
        next
      }

      arimaFit = tryCatch(arima(pricesOffset, order=c(p, 0, q)),  #not
ice that we set d to 0, thus not allowing for using differentials. As
it is a one day ahead forecast, we do not need the time series to be s
tationary, this would only be an issue for long periods forecasts.
                          error=function( err ) FALSE,
                          warning=function( err ) FALSE )

      if( !is.logical( arimaFit ) ) {
        current.aic <- AIC(arimaFit)
        if (current.aic < final.aic) {
          final.aic <- current.aic
          final.order <- c(p, 0, q)
          final.arima <- arima(pricesOffset, order=final.order)
        }
      } else {
        next
      }
    }
    forec3 = forecast(object = final.arima, h = 1)
    pred3 = c(pred3, forec3$mean)
    print(pred3)
```

```
  }
  acc <- accuracy(f = pred3, x = real_pr) #compute the accuracy of the
best arima model forecast.
  if(acc[4] < best.accuracy) {  #we will use the MPE as the measure of
accuracy
    best.accuracy <- acc[4]
    best.window <- i #this way we find the window for which MPE is the
minimum.
    print(best.window)
  } else {
    next
  }
} #in my computer (MacOs Sierra 10.12.6) this function runs for more t
han 5 days until arriving at a result;
#do not recommend to use it in yours;
#it seems that the best window is actually the minimum window: 400 day
s, with which it has a very accurate result.

#best window = 400
#best accuracy = 0.8554015 (MPE)
#for knowing the best arima we have to run the code with the optimal w
indow: 400.

btc <- read_csv('btc_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

eth <- read_csv('eth_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

btc <- btc[ , 2:ncol(btc)]
eth <- eth[ , 2:ncol(eth)]

colnames(btc) <- c("date", "close", "high", "low", "open", "volfrom",
```

```
"volto")
colnames(eth) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")

library(dplyr)
btc$date <- as.Date(btc$date)
eth$date <- as.Date(eth$date)
btc1 <- btc %>%
  filter(date > "2013-05-01")

eth1 <- eth %>%
  filter(date > "2016-05-01" )

btc1 <- btc1[1214:nrow(btc1), ] #this way, the first predicted value (
taking window = 400 into account) corresponds to row 1614, which is 20
17-10-01.

real_pr_btc <- btc1$close[windowLength:nrow(btc1)]

write.csv(real_pr_btc, "prices_btc.csv")

#518 has to be the first row: day 2017-10-01; for that reason, and tak
ing the window (400) into account, data from row 118 for eth is taken.

eth1 <- eth1[118: nrow(eth1), ]
real_pr_eth <- eth1$close[windowLength:nrow(eth1)]

write.csv(real_pr_eth, "prices_eth.csv")

library(forecast)
library(rugarch)


mean_val = c()
results1 = c()
results2 = c()

windowLength = 400
foreLength = length(btc1$close) - windowLength
real_pr <- btc1$close[windowLength:nrow(btc1)]

for (d in 0:foreLength) {

  train = btc1$close[(1+d):(windowLength+d)]

  m1.mean.model <- auto.arima(train, allowmean= T, optim.control = lis
t(maxit = 100000))

  ar.comp <- arimaorder(m1.mean.model)[1]

  d <- arimaorder(m1.mean.model)[2]

  ma.comp <- arimaorder(m1.mean.model)[3]
```

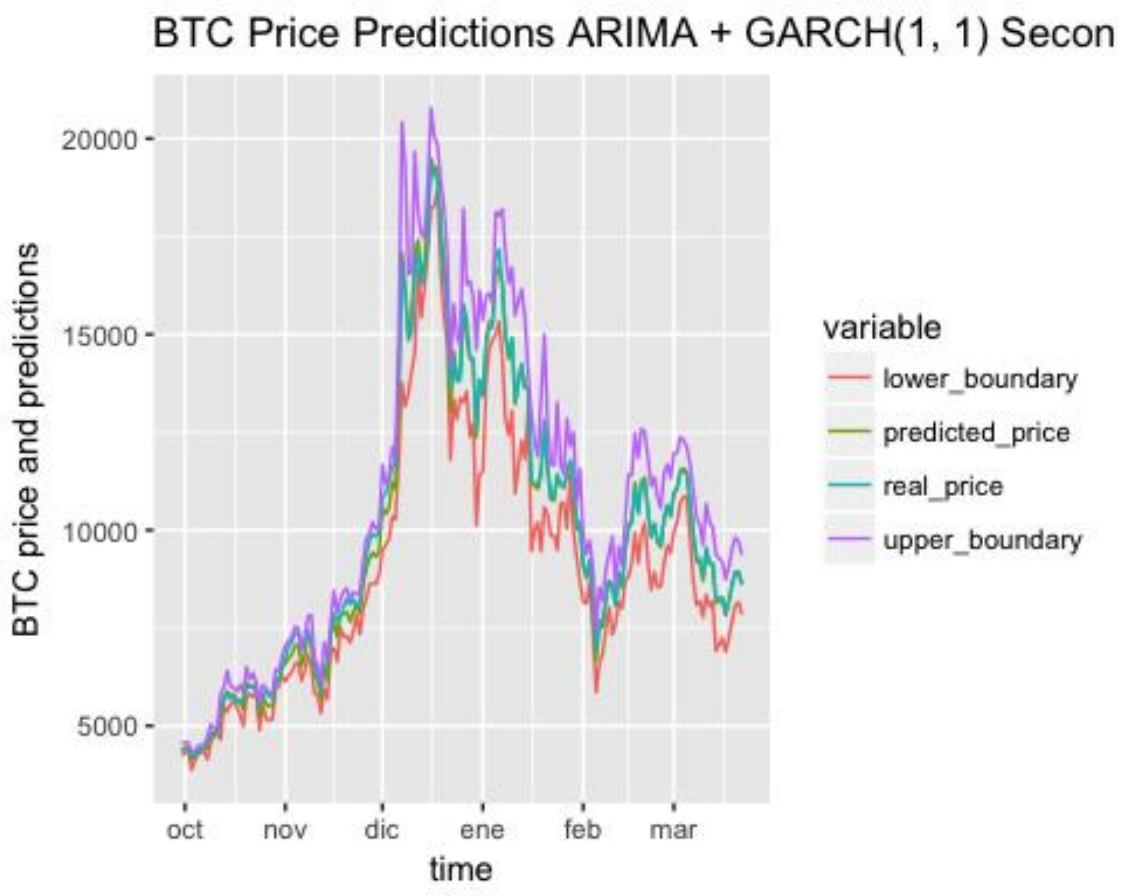
```
  model.garch = ugarchspec(mean.model = list(armaOrder=c(ar.comp,d, ma
.comp), arfima = TRUE),
                           variance.model = list(garchOrder=c(1,1)),
                           distribution.model = "std")

  model.garch.fit = ugarchfit(data = train, spec = model.garch, solver
= 'hybrid' )

  modelfor = ugarchforecast(model.garch.fit, data = NULL, n.ahead = 1)

  mean_val = c(mean_val, modelfor@forecast$seriesFor[1,1])

  results1 <- c(results1, modelfor@forecast$seriesFor[1,1] + modelfor@
forecast$sigmaFor[1,1])

  results2 <- c(results2, modelfor@forecast$seriesFor[1,1] - modelfor@
forecast$sigmaFor[1,1])
}



garch_btc_df = data.frame(cbind(btc1[400:nrow(btc1), ], mean_val, resu
lts1, results2))

colnames(garch_btc_df) <- c("date", "close", "high", "low", "open", "v
olfrom", "volto", "point_pred", "below", "above")

library(ggplot2)

ggplot(data = garch_btc_df, aes(x = date)) +
  geom_line(aes(y = close)) +
  geom_line(aes(y = point_pred, color = "red")) +
  geom_line(aes(y = below, color = "blue")) +
  geom_line(aes(y = above, color = "blue"))
```


```
acc <- accuracy(f = garch_btc_df$point_pred, x = garch_btc_df$close)

print(paste("error in terms of MAPE for btc is", acc[5]))

## [1] "error in terms of MAPE for btc is 1.60475604191107"

print(paste("accuracy for btc using rugarch package forecast is", 100-
acc[5], "%")) #98.3952439580889 %"

## [1] "accuracy for btc using rugarch package forecast is 98.39524395
80889 %"

write.csv(mean_val, file = "btc_pred_arimagarch.csv")
```

As it is seen when running the code, the time required by R to get the predictions is much higher than the one required by the LSTM model, which can be a drawback in terms of energy and time consumption. Moreover, the ARIMA + GARCH(1, 1) does not stop learning, as it does not have a learning and test period; therefore it has an advantage over LSTM, which stops learning in the training period.

```
close_prices <- btc1$close[400:nrow(btc1)]
x <- c(btc1$date[400:nrow(btc1)], btc1$date[400:nrow(btc1)], btc1$date
[400:nrow(btc1)], btc1$date[400:nrow(btc1)])
variable <- c(rep("real_price", 174), rep("predicted_price", 174), rep
("lower_boundary", 174), rep("upper_boundary", 174))
value <- c(close_prices, mean_val, results2, results1)
btc_pred <- data.frame(x, variable, value)
```

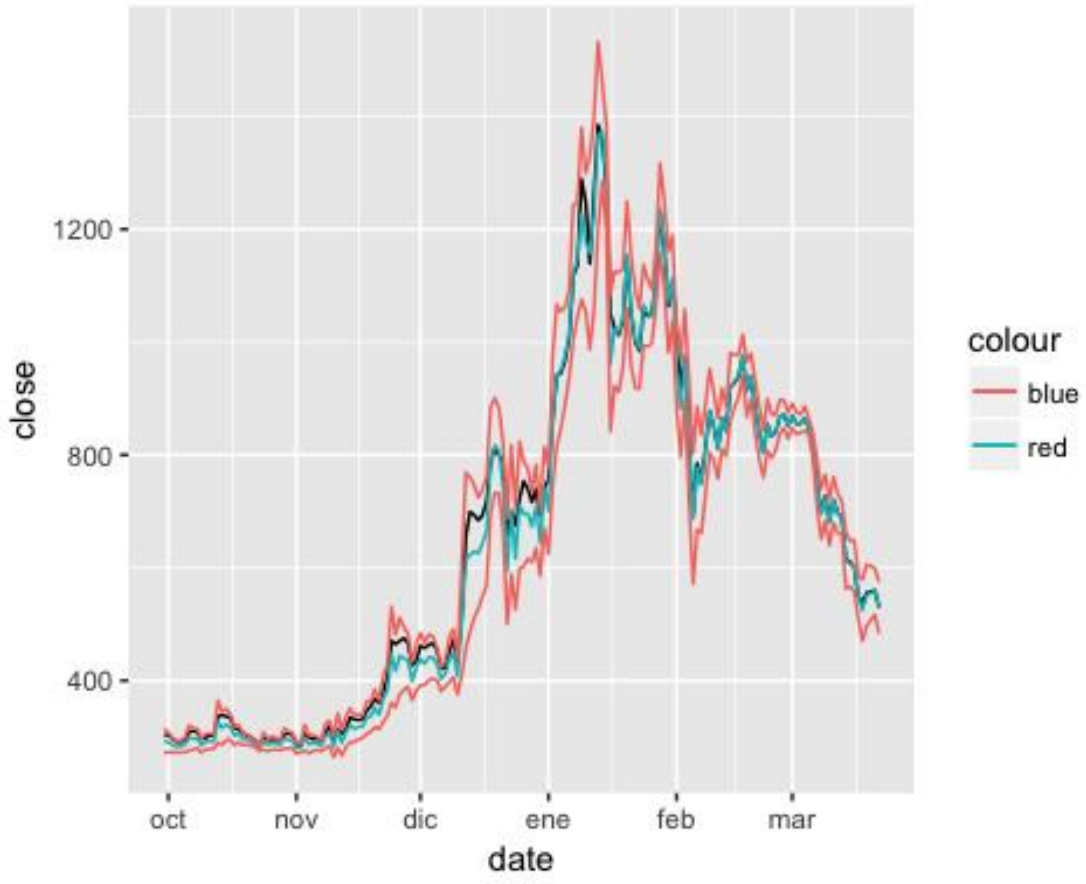
```
library(ggplot2)

btc_arima_garch_plot <- ggplot(btc_pred, aes(x = x, y = value, group =
variable, colour = variable)) +
  geom_line() +
  xlab("time") +
  ylab("BTC price and predictions") +
  ggtitle("BTC Price Predictions ARIMA + GARCH(1, 1) Second Period")

btc_arima_garch_plot
```



******by setting a concrete window, we also avoid the possible convergence problem that as R's warning states, can happen in the first ARIMA + GARCH (1,1) approach. Following the problems in https://www.queryoverflow.gdn/query/r-ugarchfit-is-there-a-convergence-indicator-20_310268.html and https://stats.stackexchange.com/questions/231280/garch1-1-fails-to-converge-in-rugarch-in-r it seems that this problem can be due to the long series used for fitting the model. For understanding deeply the problem that this poses for model forecast and the actual and concrete causes for it, the seed code for package rugarch should be revisited. *******

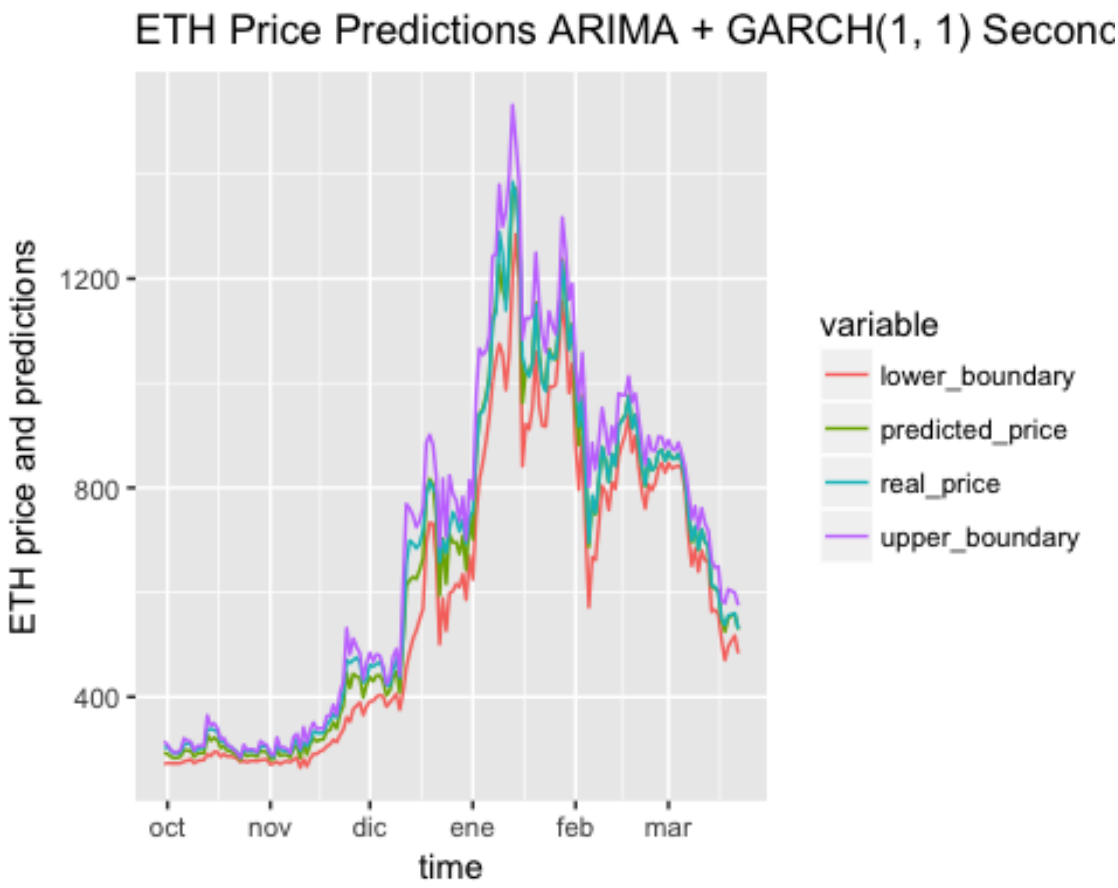
```
                                ARIMA + GARCH(1, 1) WITH ETHER


mean_val = c()
results1 = c()
results2 = c()
```

```
library(rugarch)

windowLength = 400
foreLength = length(eth1$close) - windowLength

for (d in 0:foreLength) {

  train = eth1$close[(1+d):(windowLength+d)]

  m1.mean.model <- auto.arima(train, allowmean= T, optim.control = lis
t(maxit = 100000))

  ar.comp <- arimaorder(m1.mean.model)[1]

  d <- arimaorder(m1.mean.model)[2]

  ma.comp <- arimaorder(m1.mean.model)[3]

  model.garch = ugarchspec(mean.model = list(armaOrder=c(ar.comp,d, ma
.comp), arfima = TRUE),
                           variance.model = list(garchOrder=c(1,1)),
                           distribution.model = "std")

  model.garch.fit = ugarchfit(data = train, spec = model.garch, solver
= 'hybrid' )

  modelfor = ugarchforecast(model.garch.fit, data = NULL, n.ahead = 1)

  mean_val = c(mean_val, modelfor@forecast$seriesFor[1,1])

  results1 <- c(results1, modelfor@forecast$seriesFor[1,1] + modelfor@
forecast$sigmaFor[1,1])

  results2 <- c(results2, modelfor@forecast$seriesFor[1,1] - modelfor@
forecast$sigmaFor[1,1])
}



garch_btc_df = data.frame(cbind(eth1[400:nrow(eth1), ], mean_val, resu
lts1, results2))

colnames(garch_btc_df) <- c("date", "close", "high", "low", "open", "v
olfrom", "volto", "point_pred", "below", "above")

library(ggplot2)

ggplot(data = garch_btc_df, aes(x = date)) +
  geom_line(aes(y = close)) +
  geom_line(aes(y = point_pred, color = "red")) +
  geom_line(aes(y = below, color = "blue")) +
  geom_line(aes(y = above, color = "blue"))
```




```
acc <- accuracy(f = garch_btc_df$point_pred, x = garch_btc_df$close)

print(paste("error in terms of MAPE for eth is", acc[5]))

## [1] "error in terms of MAPE for eth is 2.62516416008843"

print(paste("accuracy for eth using rugarch package forecast is", 100-
acc[5], "%"))

## [1] "accuracy for eth using rugarch package forecast is 97.37483583
99116 %"

#97.3748358399116 %


write.csv(mean_val, file = "eth_pred_arimagarch.csv")

close_prices <- eth1$close[400:nrow(eth1)]
x <- c(eth1$date[400:nrow(eth1)], eth1$date[400:nrow(eth1)], eth1$date
[400:nrow(eth1)], eth1$date[400:nrow(eth1)]) #change the 173 for the l
ength of the prediction you'll get.
variable <- c(rep("real_price", 174), rep("predicted_price", 174), rep
("lower_boundary", 174), rep("upper_boundary", 174))
value <- c(close_prices, mean_val, results2, results1)
eth_pred <- data.frame(x, variable, value)

library(ggplot2)
```

```
ggplot(eth_pred, aes(x = x, y = value, group = variable, colour = vari
able)) +
  geom_line() +
  xlab("time") +
  ylab("ETH price and predictions") +
  ggtitle("ETH Price Predictions ARIMA + GARCH(1, 1) Second Period")
```



ANNEX 4

The full code for this Annex is in the following Github Repository. I uploaded it there so that replication and improvement can be easily carried out:

ANNEX 4 (1): LSTM (1):

https://github.com/alexvaca0/TFG18/blob/master/LSTM(1)_DEFINITIVO.ipynb

ANNEX 4 (2): LSTM (2):

https://github.com/alexvaca0/TFG18/blob/master/LSTM(2)_DEFINITIVO.ipynb

ANNEX 4 (3): LSTM (3) for bitcoin:

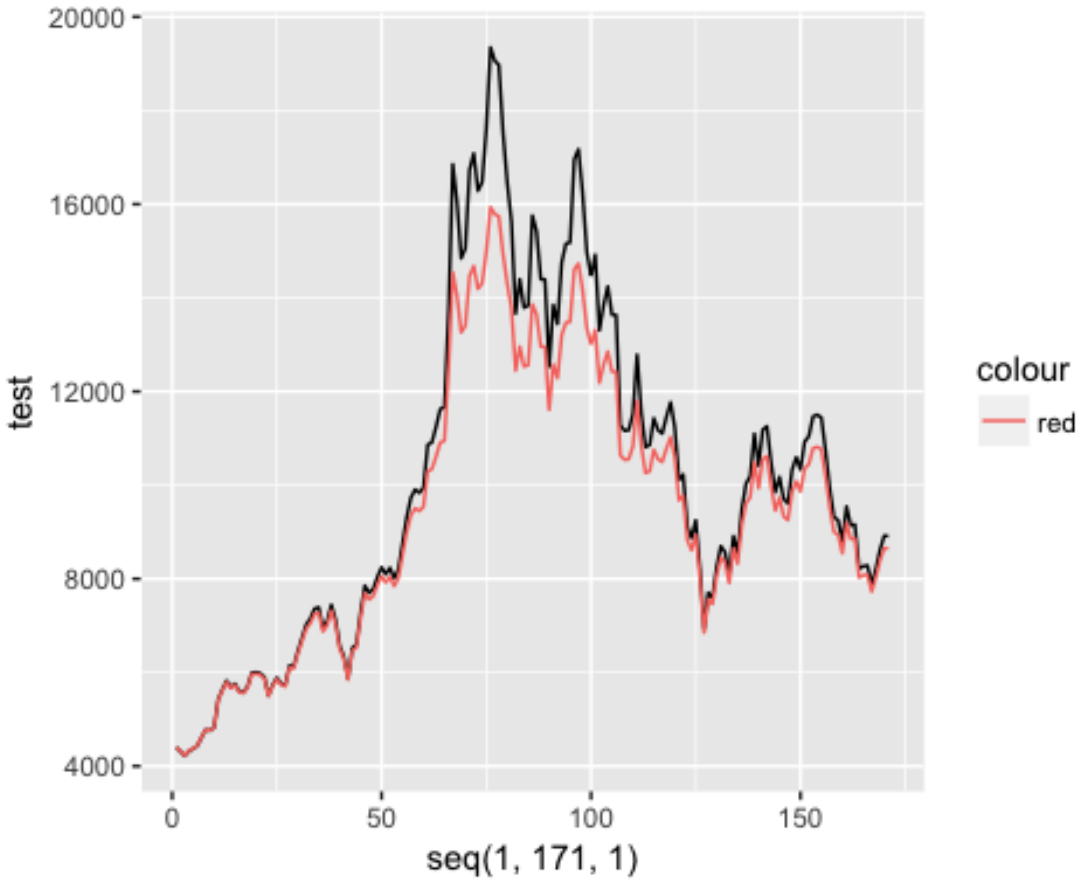
https://github.com/alexvaca0/TFG18/blob/master/LSTM(3)_BTC.ipynb

ANNEX 4 (4): LSTM (3) for ether:

https://github.com/alexvaca0/TFG18/blob/master/LSTM(3)_ETH.ipynb

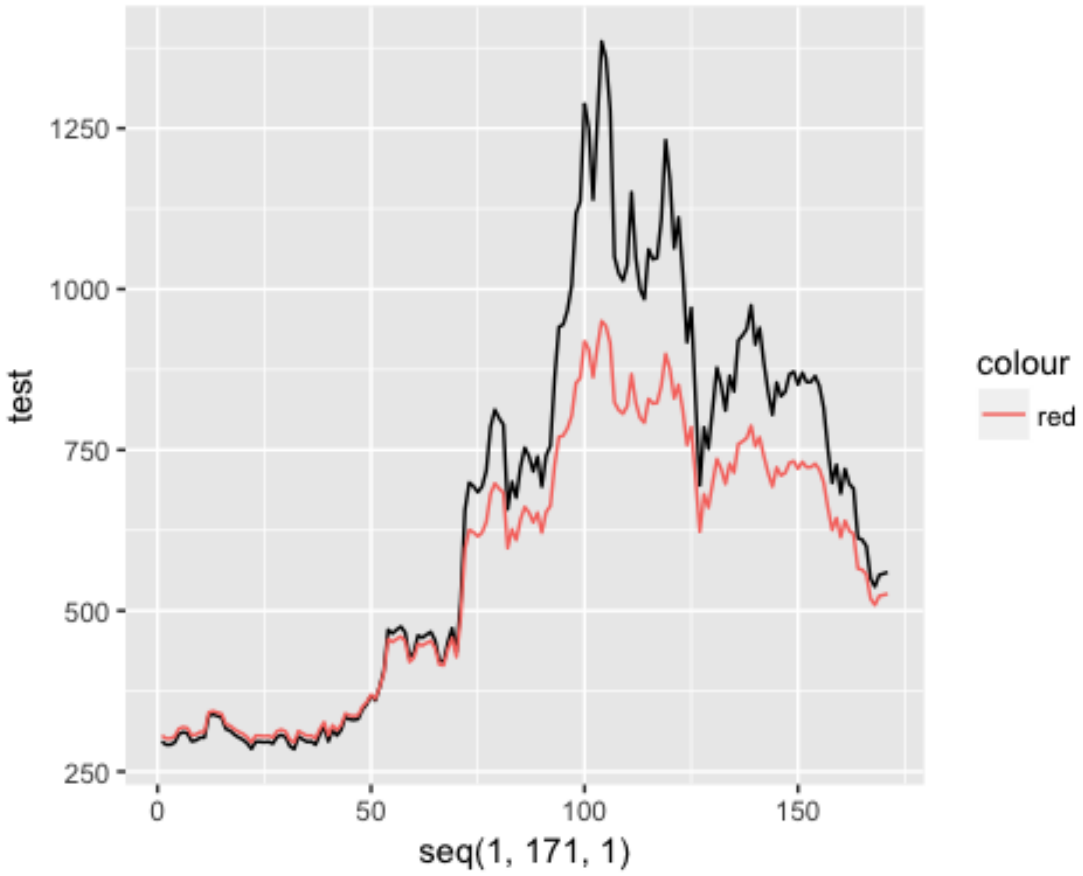
```
library(readr)

p <- read_csv("btc_prediction2_prueba.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

predicted <- p

test <- read_csv("btc_test2_prueba.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

df <- cbind(predicted, test)

colnames(df) = c("pred", "test")

library(ggplot2)
library(forecast)

df <- df[-nrow(df), ]
```

```
ggplot(data = df, aes(x = seq(1, 171, 1))) +
  geom_line(aes(y = test)) +
  geom_line(aes(y = pred, color = "red"))
```




```
#we need to take the last observation out, it's a clear outlier and di
sturbs any measurement.



acc <- accuracy(f = df$pred, x = df$test)

print(paste("% accuracy:", 100 - acc[5], "%"))

## [1] "% accuracy: 95.1009347340747 %"

library(readr)

p <- read_csv("eth_prediction2_prueba.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

predicted <- p
```

```
test <- read_csv("eth_test2_prueba.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

df <- cbind(predicted, test)

colnames(df) = c("pred", "test")

library(ggplot2)
library(forecast)

df <- df[-nrow(df), ]

ggplot(data = df, aes(x = seq(1, 171, 1))) +
  geom_line(aes(y = test)) +
  geom_line(aes(y = pred, color = "red"))
```




```
#we need to take the last observation out, it's a clear outlier and di
sturbs any measurement.



acc <- accuracy(f = df$pred, x = df$test)
```

```r
print(paste("% accuracy:", 100 - acc[5], "%"))
```

```
## [1] "% accuracy: 89.6271599279923 %"
```

ANNEX 5: TESTING STRATEGIES

In this Annex, the objective will be to test the strategies and see which model provides a higher return, as well as to analyze the dependence of each algorithm on the market. For comparing the performance of both models in this respect, we should use predictions starting in 2017-10-07, as it is the first predicted value we have with the LSTM models.

```
library(readr)
btc <- read_csv("btc_usd_22_03_18.csv")

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

eth <- read_csv("eth_usd_22_03_18.csv")

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

library(readr)

btc <- read_csv('btc_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )
```

```r
eth <- read_csv('eth_usd_22_03_18.csv')

## Parsed with column specification:
## cols(
##   time = col_double(),
##   timeDate = col_date(format = ""),
##   close = col_double(),
##   high = col_double(),
##   low = col_double(),
##   open = col_double(),
##   volumefrom = col_double(),
##   volumeto = col_double()
## )

btc <- btc[ , 2:ncol(btc)]
eth <- eth[ , 2:ncol(eth)]

colnames(btc) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")
colnames(eth) <- c("date", "close", "high", "low", "open", "volfrom",
"volto")

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

btc$date <- as.Date(btc$date)
eth$date <- as.Date(eth$date)

btc <- btc %>%
  filter(date > "2017-09-30")

eth <- eth %>%
  filter(date > "2017-09-30")

lstm_1_btc <- read_csv("btc_pred_prices_lstm_1_def.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

lstm_1_eth <- read_csv("eth_pred_prices_lstm_1_def.csv",
col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

head(lstm_1_btc)

## # A tibble: 6 x 1
##      X1
##   <dbl>
## 1 4353.
## 2 4393.
## 3 4485.
## 4 4663.
## 5 4782.
## 6 4779.

colnames(lstm_1_btc) <- c("btc_pred_lstm_1")
colnames(lstm_1_eth) <- c("eth_pred_lstm_1")

btc_pred_arimagarch <- read_csv("btc_pred_arimagarch.csv")

## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   x = col_double()
## )

eth_pred_arimagarch <- read_csv("eth_pred_arimagarch.csv")

## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   x = col_double()
## )

nrow(lstm_1_btc) #167

## [1] 167

nrow(lstm_1_eth) #167

## [1] 167

nrow(btc_pred_arimagarch) #174

## [1] 174

nrow(eth_pred_arimagarch) #174

## [1] 174

btc_pred_arimagarch <- btc_pred_arimagarch[ ,2]
eth_pred_arimagarch <- eth_pred_arimagarch[, 2]
```

```r
colnames(btc_pred_arimagarch) <- c("btc_pred_AG")
colnames(eth_pred_arimagarch) <- c("eth_pred_AG")

lstm_2_btc <- read_csv("btc_pred_prices_lstm_2_def.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

lstm_2_eth <- read_csv("eth_pred_prices_lstm_2_def.csv",
col_names = FALSE)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

head(lstm_2_btc)

## # A tibble: 6 x 1
##      X1
##   <dbl>
## 1 4305.
## 2 4343.
## 3 4462.
## 4 4675.
## 5 4785.
## 6 4774.

colnames(lstm_2_btc) <- c("btc_pred_lstm_2")
colnames(lstm_2_eth) <- c("eth_pred_lstm_2")

lstm_3_btc <- read_csv("btc_prediction2_prueba.csv", col_names = FALSE
)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

colnames(lstm_3_btc) = c("btc_lstm3")


lstm_3_eth <- read_csv("eth_prediction2_prueba.csv", col_names = FALSE
)

## Parsed with column specification:
## cols(
##   X1 = col_double()
## )

colnames(lstm_3_eth) = c("eth_lstm3")
head(lstm_3_eth)
```

```
## # A tibble: 6 x 1
##   eth_lstm3
##       <dbl>
## 1      306.
## 2      301.
## 3      301.
## 4      304.
## 5      317.
## 6      319.
```

```
btc_pred_arimagarch <- btc_pred_arimagarch[(174-166):174, ]
nrow(btc_pred_arimagarch)
```

```
## [1] 167
```

```
eth_pred_arimagarch <- eth_pred_arimagarch[(174-166):174, ]

lstm_3_btc <- lstm_3_btc[(172-166):172, ]

lstm_3_eth <- lstm_3_eth[(172-166):172, ]



df <- cbind(btc_pred_arimagarch, eth_pred_arimagarch, lstm_1_btc, lstm
_1_eth, lstm_2_btc, lstm_2_eth, lstm_3_btc, lstm_3_eth)

btc$return <- c(NA, diff(log(btc$close)))

btc <- btc %>%
  filter(date > "2017-10-06")
nrow(btc)
```

```
## [1] 167
```

```
eth$return <- c(NA, diff(log(eth$close)))

eth <- eth %>%
  filter(date > "2017-10-06")

nrow(eth)
```

```
## [1] 167
```

```
df$btc_return <- btc$return
df$eth_return <- eth$return

df$date <- as.Date(btc$date)

df$btc_price <- btc$close
df$eth_price <- eth$close

head(df)
```

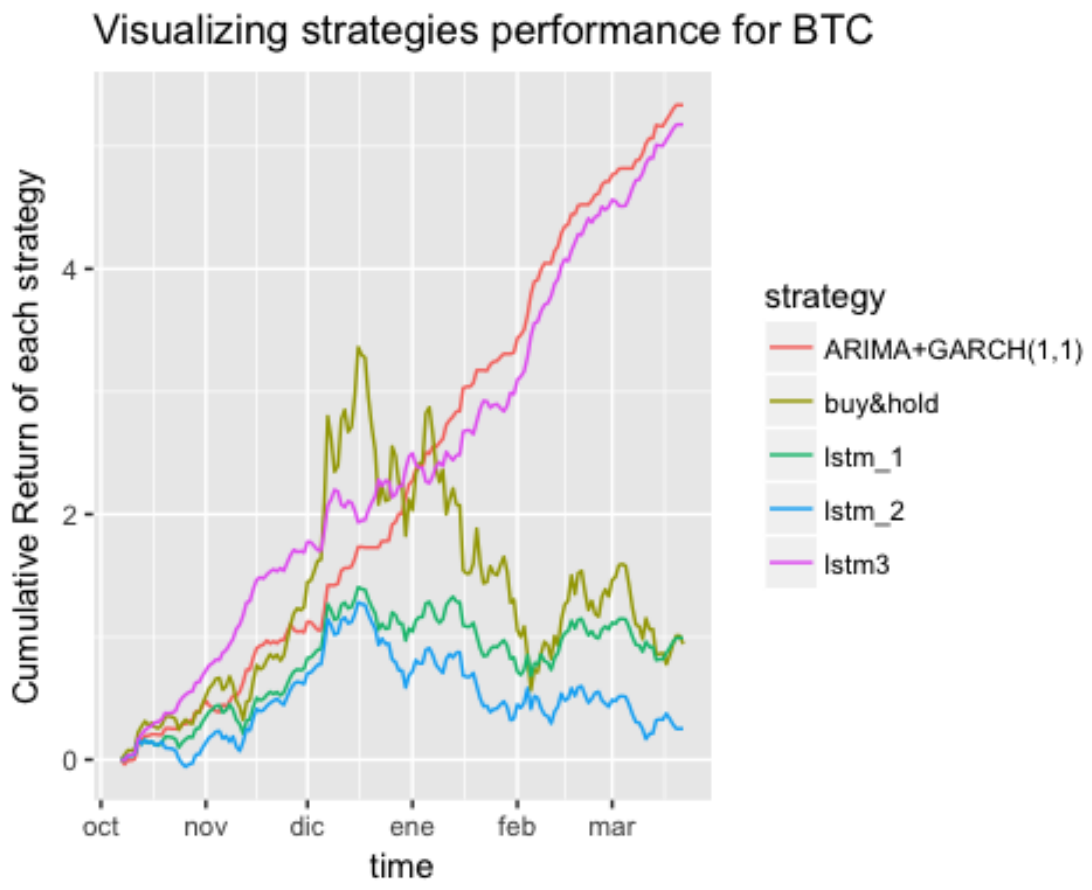
```
##   btc_pred_AG eth_pred_AG btc_pred_lstm_1 eth_pred_lstm_1 btc_pred_
lstm_2
## 1    4430.717    297.1314        4353.146        296.5374        43
```

```
04.905
## 2    4433.966    297.4039        4393.272        304.6655        43
42.981
## 3    4817.990    285.2017        4485.116        306.2822        44
62.151
## 4    4820.648    291.8229        4662.664        304.6479        46
74.660
## 5    4876.863    292.6564        4782.388        295.5265        47
84.675
## 6    5230.573    292.7514        4778.568        298.4584        47
73.831
##   eth_pred_lstm_2 btc_lstm3 eth_lstm3   btc_return    eth_return
## 1        297.8919  4438.434  319.1638  0.014503391  9.457938e-03
## 2        307.0427  4612.223  317.5611  0.038886315 -5.702794e-03
## 3        308.7445  4775.523  306.1569  0.035318751 -4.136201e-02
## 4        305.5522  4761.626  307.5347 -0.002962002  5.072146e-03
## 5        296.9384  4821.441  311.5424  0.012691616  1.463473e-02
## 6        298.6796  5415.619  311.5697  0.118776604  9.905076e-05
##         date btc_price eth_price
## 1 2017-10-07   4435.81    311.26
## 2 2017-10-08   4611.70    309.49
## 3 2017-10-09   4777.49    296.95
## 4 2017-10-10   4763.36    298.46
## 5 2017-10-11   4824.20    302.86
## 6 2017-10-12   5432.62    302.89

df_btc <- df[ , c(1, 3, 5, 7, 9, 11, 12)]
#now we have to put the predicted values for both models one day befor
e they are, as we want to know today the predicted value for tomorrow,
in order to decide our position today.

df_eth <- df[ , c(2, 4, 6, 8, 10, 11, 13)]

df_btc$btc_pred_AG <- c(df_btc$btc_pred_AG[2:nrow(df_btc)], NA)

df_btc$btc_pred_lstm_1 <- c(df_btc$btc_pred_lstm_1 [3:nrow(df_btc)], N
A, NA)

df_btc$btc_pred_lstm_2 <- c(df_btc$btc_pred_lstm_2[3:nrow(df_btc)], NA
, NA)

df_btc$btc_lstm3 <- c(df_btc$btc_lstm3[2:nrow(df_btc)], NA)

pos_AG <- c()

for(i in 1:(nrow(df_btc)-1)) {
  if(df_btc$btc_pred_AG[i] > df_btc$btc_price[i]) { #if the prediction
today, for tomorrow's value, says that the price tomorrow will be high
er than the price today...
      pos_AG <- c(pos_AG, 1)
  } else if (df$btc_pred_AG[i] < df_btc$btc_price[i]) {
      pos_AG <- c(pos_AG, -1)
  } else {
      pos_AG <- c(pos_AG, 0)
  }
```

```
}

length(pos_AG)

## [1] 166

df_btc$pos_AG <- c(pos_AG, NA)

cum_ret_AG <- c(0)

for (i in 2:nrow(df_btc)) {
  ret_i <- df_btc$btc_return[i]*df_btc$pos_AG[i - 1]
  cumret_i <- cum_ret_AG[i - 1] + ret_i
  cum_ret_AG <- c(cum_ret_AG, cumret_i)
}

df_btc$cumret_AG <- cum_ret_AG

pos_lstm_1 <- c()

for(i in 2:(nrow(df_btc)-2)) {
  if(df_btc$btc_pred_lstm_1[i] > df_btc$btc_price[i]) {
      pos_lstm_1 <- c(pos_lstm_1, 1)
  } else if (df$btc_pred_lstm_1[i] < df_btc$btc_price[i]) {
      pos_lstm_1 <- c(pos_lstm_1, -1)
  } else {
      pos_lstm_1 <- c(pos_lstm_1, -1)
  }
}

length(pos_lstm_1)

## [1] 164

df_btc$pos_lstm_1 <- c(0, pos_lstm_1, 0, 0)

cum_ret_lstm_1 <- c(0)

for (i in 2:nrow(df_btc)) {
  ret_i <- df_btc$btc_return[i]*df_btc$pos_lstm_1[i - 1]
  cumret_i <- cum_ret_lstm_1[i - 1] + ret_i
  cum_ret_lstm_1 <- c(cum_ret_lstm_1, cumret_i)
}

df_btc$cumret_lstm_1 <- cum_ret_lstm_1

pos_lstm_2 <- c()

for(i in 2:(nrow(df_btc)-2)) {
  if(df_btc$btc_pred_lstm_2[i] > df_btc$btc_price[i]) {
      pos_lstm_2 <- c(pos_lstm_2, 1)
  } else if (df$btc_pred_lstm_2[i] < df_btc$btc_price[i]) {
      pos_lstm_2 <- c(pos_lstm_2, -1)
  } else {
      pos_lstm_2 <- c(pos_lstm_2, -1)
  }
```

```
}

length(pos_lstm_2)

## [1] 164

df_btc$pos_lstm_2 <- c(0, pos_lstm_2, 0, 0)

cum_ret_lstm_2 <- c(0)

for (i in 2:nrow(df_btc)) {
  ret_i <- df_btc$btc_return[i]*df_btc$pos_lstm_2[i - 1]
  cumret_i <- cum_ret_lstm_2[i - 1] + ret_i
  cum_ret_lstm_2 <- c(cum_ret_lstm_2, cumret_i)
}

df_btc$cumret_lstm_2 <- cum_ret_lstm_2

cum_ret_buyhold <- c(0)

for (i in 2:nrow(df_btc)) {
  first_price <- df_btc$btc_price[1]
  cum_ret_until_today <- (df_btc$btc_price[i] - first_price) / first_p
rice
  cum_ret_buyhold <- c(cum_ret_buyhold, cum_ret_until_today)
}

df_btc$cumret_buyhold <- cum_ret_buyhold

pos_lstm3 <- c()

for(i in 2:(nrow(df_btc)-2)) {
  if(df_btc$btc_lstm3[i] > df_btc$btc_price[i] ) { #| df_btc$lstm3[i]
> df_btc$lstm3[i-1]) {
      pos_lstm3 <- c(pos_lstm3, 1)
  } else if (df_btc$btc_lstm3[i] < df_btc$btc_price[i]) {  #| df_btc$l
stm3[i] < df_btc$lstm3[i-1]) {
      pos_lstm3 <- c(pos_lstm3, -1)
  } else {
      pos_lstm3 <- c(pos_lstm3, -1)
  }
}

length(pos_lstm3)

## [1] 164

df_btc$pos_lstm3 <- c(0, pos_lstm3, 0, 0)

cum_ret_lstm3 <- c(0)

for (i in 2:nrow(df_btc)) {
  ret_i <- df_btc$btc_return[i]*df_btc$pos_lstm3[i - 1]
  cumret_i <- cum_ret_lstm3[i - 1] + ret_i
  cum_ret_lstm3 <- c(cum_ret_lstm3, cumret_i)
}
```

```r
df_btc$cumret_lstm3 <- cum_ret_lstm3

x <- c(df_btc$date, df_btc$date, df_btc$date, df_btc$date, df_btc$date
)
strategy <- c(rep("buy&hold", 167), rep("ARIMA+GARCH(1,1)", 167), rep(
"lstm_1", 167), rep("lstm_2", 167), rep("lstm3", 167))

value <- c(df_btc$cumret_buyhold, df_btc$cumret_AG, df_btc$cumret_lstm
_1, df_btc$cumret_lstm_2, df_btc$cumret_lstm3)

btc_df_plot <- data.frame(x, strategy, value)


library(ggplot2)

set.seed(717)

ggplot(btc_df_plot, aes(x = x, y = value, group = strategy, colour = s
trategy)) +
  geom_line() +
  xlab("time") +
  ylab("Cumulative Return of each strategy") +
  ggtitle("Visualizing strategies performance for BTC")
```

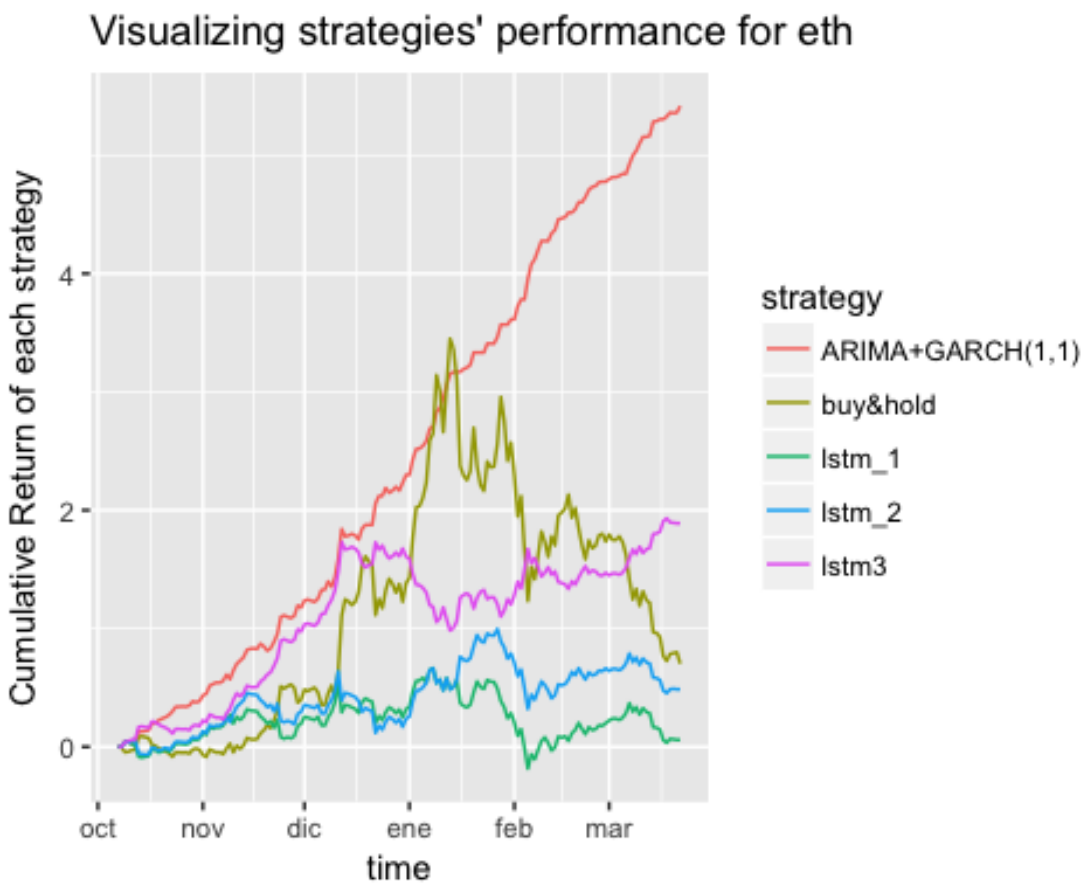


The only strategy for bitcoin that clearly outperforms the Buy & Hold is the ARIMA + GARCH (1, 1). Now, the same has to be done for ether.

```r
df_eth$eth_pred_AG <- c(df_eth$eth_pred_AG[2:nrow(df_eth)], NA)

df_eth$eth_pred_lstm_1 <- c(df_eth$eth_pred_lstm_1 [3:nrow(df_eth)], N
A, NA)

df_eth$eth_pred_lstm_2 <- c(df_eth$eth_pred_lstm_2[3:nrow(df_eth)], NA
, NA)

df_eth$eth_lstm3 <-c(df_eth$eth_lstm3[2:nrow(df_btc)], NA)

pos_AG <- c()

for(i in 1:(nrow(df_eth)-1)) {
  if(df_eth$eth_pred_AG[i] > df_eth$eth_price[i]) { #if the prediction
today, for tomorrow's value, says that the price tomorrow will be high
er than the price today...
      pos_AG <- c(pos_AG, 1)
  } else if (df$eth_pred_AG[i] < df_eth$eth_price[i]) {
      pos_AG <- c(pos_AG, -1)
  } else {
      pos_AG <- c(pos_AG, 0)
  }
}

length(pos_AG)

## [1] 166

df_eth$pos_AG <- c(pos_AG, NA)

cum_ret_AG <- c(0)

for (i in 2:nrow(df_eth)) {
  ret_i <- df_eth$eth_return[i]*df_eth$pos_AG[i - 1]
  cumret_i <- cum_ret_AG[i - 1] + ret_i
  cum_ret_AG <- c(cum_ret_AG, cumret_i)
}

df_eth$cumret_AG <- cum_ret_AG

pos_lstm_1 <- c()

for(i in 2:(nrow(df_eth)-2)) {
  if(df_eth$eth_pred_lstm_1[i] > df_eth$eth_price[i]) {
      pos_lstm_1 <- c(pos_lstm_1, 1)
  } else if (df$eth_pred_lstm_1[i] < df_eth$eth_price[i]) {
      pos_lstm_1 <- c(pos_lstm_1, -1)
  } else {
      pos_lstm_1 <- c(pos_lstm_1, -1)
  }
}

length(pos_lstm_1)

## [1] 164
```

```r
df_eth$pos_lstm_1 <- c(0, pos_lstm_1, 0, 0)

cum_ret_lstm_1 <- c(0)

for (i in 2:nrow(df_eth)) {
  ret_i <- df_eth$eth_return[i]*df_eth$pos_lstm_1[i - 1]
  cumret_i <- cum_ret_lstm_1[i - 1] + ret_i
  cum_ret_lstm_1 <- c(cum_ret_lstm_1, cumret_i)
}

df_eth$cumret_lstm_1 <- cum_ret_lstm_1

pos_lstm_2 <- c()

for(i in 2:(nrow(df_eth)-2)) {
  if(df_eth$eth_pred_lstm_2[i] > df_eth$eth_price[i]) {
      pos_lstm_2 <- c(pos_lstm_2, 1)
  } else if (df$eth_pred_lstm_2[i] < df_eth$eth_price[i]) {
      pos_lstm_2 <- c(pos_lstm_2, -1)
  } else {
      pos_lstm_2 <- c(pos_lstm_2, -1)
  }
}

length(pos_lstm_2)

## [1] 164

df_eth$pos_lstm_2 <- c(0, pos_lstm_2, 0, 0)

cum_ret_lstm_2 <- c(0)

for (i in 2:nrow(df_eth)) {
  ret_i <- df_eth$eth_return[i]*df_eth$pos_lstm_2[i - 1]
  cumret_i <- cum_ret_lstm_2[i - 1] + ret_i
  cum_ret_lstm_2 <- c(cum_ret_lstm_2, cumret_i)
}

df_eth$cumret_lstm_2 <- cum_ret_lstm_2

cum_ret_buyhold <- c(0)

for (i in 2:nrow(df_eth)) {
  first_price <- df_eth$eth_price[1]
  cum_ret_until_today <- (df_eth$eth_price[i] - first_price) / first_p
rice
  cum_ret_buyhold <- c(cum_ret_buyhold, cum_ret_until_today)
}

df_eth$cumret_buyhold <- cum_ret_buyhold

pos_lstm_3 <- c()

for(i in 2:(nrow(df_eth)-2)) {
  if(df_eth$eth_lstm3[i] > df_eth$eth_price[i]) {
      pos_lstm_3 <- c(pos_lstm_3, 1)
```

```
  } else if (df$eth_lstm3[i] < df_eth$eth_price[i]) {
      pos_lstm_3 <- c(pos_lstm_3, -1)
  } else {
      pos_lstm_3 <- c(pos_lstm_3, -1)
  }
}

length(pos_lstm_3)

## [1] 164

df_eth$pos_lstm_3 <- c(0, pos_lstm_3, 0, 0)

cum_ret_lstm_3 <- c(0)

for (i in 2:nrow(df_eth)) {
  ret_i <- df_eth$eth_return[i]*df_eth$pos_lstm_3[i - 1]
  cumret_i <- cum_ret_lstm_3[i - 1] + ret_i
  cum_ret_lstm_3 <- c(cum_ret_lstm_3, cumret_i)
}

df_eth$cumret_lstm_3 <- cum_ret_lstm_3

x <- c(df_eth$date, df_eth$date, df_eth$date, df_eth$date, df_eth$date
)
strategy <- c(rep("buy&hold", 167), rep("ARIMA+GARCH(1,1)", 167), rep(
"lstm_1", 167), rep("lstm_2", 167), rep("lstm3", 167))
value <- c(df_eth$cumret_buyhold, df_eth$cumret_AG, df_eth$cumret_lstm
_1, df_eth$cumret_lstm_2, df_eth$cumret_lstm_3)

eth_df_plot <- data.frame(x, strategy, value)

library(ggplot2)

ggplot(eth_df_plot, aes(x = x, y = value, group = strategy, colour = s
trategy)) +
  geom_line() +
  xlab("time") +
  ylab("Cumulative Return of each strategy") +
  ggtitle("Visualizing strategies' performance for eth")
```



Now, get information for building the model summary table.

```
cum_ret_btc_AG <- df_btc$cumret_AG[nrow(df_btc)]
cum_ret_eth_AG <- df_eth$cumret_AG[nrow(df_eth)]
cum_ret_btc_lstm_1 <- df_btc$cumret_lstm_1[nrow(df_btc)]
cum_ret_eth_lstm_1 <- df_eth$cumret_lstm_1[nrow(df_eth)]
cum_ret_btc_lstm_2 <- df_btc$cumret_lstm_2[nrow(df_btc)]
cum_ret_eth_lstm_2 <- df_eth$cumret_lstm_2[nrow(df_eth)]
cum_ret_btc_buyhold <- df_btc$cumret_buyhold[nrow(df_btc)]
cum_ret_eth_buyhold <- df_eth$cumret_buyhold[nrow(df_eth)]
cum_ret_btc_lstm_3 <- df_btc$cumret_lstm3[nrow(df_btc)]
cum_ret_eth_lstm_3 <- df_eth$cumret_lstm_3[nrow(df_eth)]
#cum_ret_btc_AG
#cum_ret_eth_AG
#cum_ret_btc_lstm_1
#cum_ret_eth_lstm_1
#cum_ret_btc_lstm_2
#cum_ret_eth_lstm_2
#cum_ret_btc_buyhold
#cum_ret_eth_buyhold
#cum_ret_btc_lstm_3
#cum_ret_eth_lstm_3

#for seeing the results, just take # out below.

df_btc$profit_AG <- 1000*(1 + df_btc$cumret_AG) - 1000
#df_btc$profit_AG
```

```
df_btc$profit_lstm1 <- 1000*(1 + df_btc$cumret_lstm_1) - 1000
#df_btc$profit_lstm1
df_btc$profit_lstm2 <- 1000*(1 + df_btc$cumret_lstm_2) - 1000
#df_btc$profit_lstm2
df_btc$profit_buyhold <- 1000*(1 + df_btc$cumret_buyhold) - 1000
#df_btc$profit_buyhold
df_btc$profit_lstm3 <- 1000*(1 + df_btc$cumret_lstm3) - 1000
#df_btc$profit_lstm3

df_eth$profit_AG <- 1000*(1 + df_eth$cumret_AG) - 1000
#df_eth$profit_AG
df_eth$profit_lstm1 <- 1000*(1 + df_eth$cumret_lstm_1) - 1000
#df_eth$profit_lstm1
df_eth$profit_lstm2 <- 1000*(1 + df_eth$cumret_lstm_2) - 1000
#df_eth$profit_lstm2
df_eth$profit_buyhold <- 1000*(1 + df_eth$cumret_buyhold) -1000
#df_eth$profit_buyhold
df_eth$profit_lstm3 <- 1000*(1 + df_eth$cumret_lstm_3) - 1000
#df_eth$profit_lstm3
```

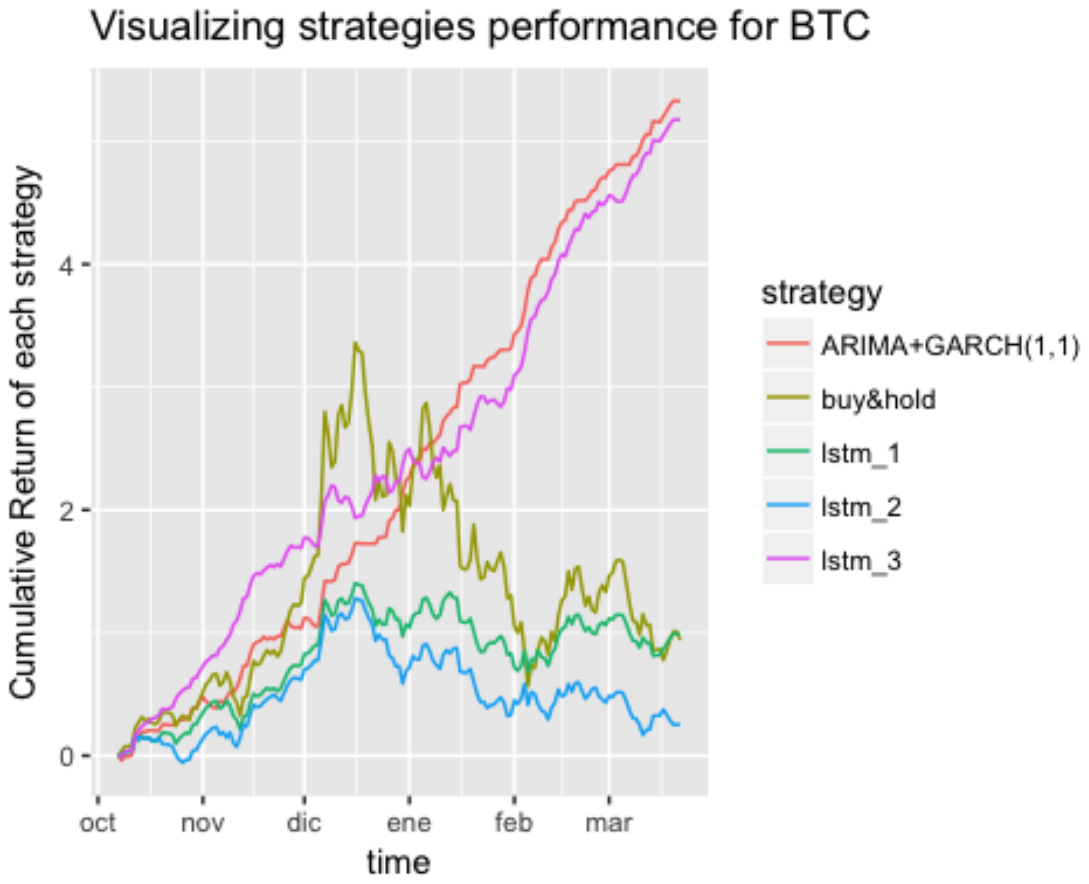
Now the fee costs will also be considered:

```
change_pos <- c()

for(i in 2:(nrow(df_btc)-1)) {
  if(df_btc$pos_AG[i] == df_btc$pos_AG[i - 1]) {
    change_pos <- c(change_pos, 0)
  } else if(df_btc$pos_AG[i] != df_btc$pos_AG[i - 1]) {
    change_pos <- c(change_pos, 1)
  }
}

df_btc$change_pos_AG <- c(NA, change_pos, NA)

df_btc$AG_costs <- df_btc$change_pos_AG*0.0005*(df_btc$profit_AG+1000)
#0.0005 is the fee cost in Binance

change_pos <- c()

for(i in 2:(nrow(df_btc)-1)) {
  if(df_btc$pos_lstm_1[i] == df_btc$pos_lstm_1[i - 1]) {
    change_pos <- c(change_pos, 0)
  } else if(df_btc$pos_lstm_1[i] != df_btc$pos_lstm_1[i - 1]) {
    change_pos <- c(change_pos, 1)
  }
}

df_btc$change_pos_l1 <- c(0, change_pos, 0)

df_btc$lstm1_costs <- df_btc$change_pos_l1*0.0005*(df_btc$profit_lstm1
+ 1000)

change_pos <- c()
```

```r
for(i in 2:(nrow(df_btc)-1)) {
  if(df_btc$pos_lstm_2[i] == df_btc$pos_lstm_2[i - 1]) {
    change_pos <- c(change_pos, 0)
  } else if(df_btc$pos_lstm_2[i] != df_btc$pos_lstm_2[i - 1]) {
    change_pos <- c(change_pos, 1)
  }
}

df_btc$change_pos_l2 <- c(0, change_pos, 0)

df_btc$lstm2_costs <- df_btc$change_pos_l2*0.0005*(df_btc$profit_lstm2
+ 1000)

change_pos <- c()

for(i in 2:(nrow(df_btc)-1)) {
  if(df_btc$pos_lstm3[i] == df_btc$pos_lstm3[i - 1]) {
    change_pos <- c(change_pos, 0)
  } else if(df_btc$pos_lstm3[i] != df_btc$pos_lstm3[i - 1]) {
    change_pos <- c(change_pos, 1)
  }
}

df_btc$change_pos_l3 <- c(0, change_pos, 0)

df_btc$lstm3_costs <- df_btc$change_pos_l3*0.0005*(df_btc$profit_lstm3
+ 1000)

df_btc2 <- df_btc

df_btc2$change_pos_AG[1] <- 0
df_btc2$change_pos_AG[nrow(df_btc2)] <- 0

cum_ret_AG <- c(0)

for (i in 2:nrow(df_btc2)) {
        if(df_btc2$change_pos_AG[i-1] == 1) {
                ret_i <- (df_btc2$btc_return[i]-0.0005)*df_btc2$pos_AG
[i - 1]
                cumret_i <- cum_ret_AG[i - 1] + ret_i
                cum_ret_AG <- c(cum_ret_AG, cumret_i)
  }     else {
                ret_i <- df_btc2$btc_return[i]*df_btc2$pos_AG[i - 1]
                cumret_i <- cum_ret_AG[i - 1] + ret_i
                cum_ret_AG <- c(cum_ret_AG, cumret_i)
  }
}


df_btc2$cumretAG <- cum_ret_AG

cum_ret_lstm_1 <- c(0)

for (i in 2:nrow(df_btc2))  {
        if(df_btc2$change_pos_l1[i - 1] == 1) {
```

```
                ret_i <- (df_btc2$btc_return[i]-0.0005)*df_btc2$pos_ls
tm_1[i - 1]
                cumret_i <- cum_ret_lstm_1[i - 1] + ret_i
                cum_ret_lstm_1 <- c(cum_ret_lstm_1, cumret_i)
  }     else {
                ret_i <- df_btc2$btc_return[i]*df_btc2$pos_lstm_1[i -
1]
                cumret_i <- cum_ret_lstm_1[i - 1] + ret_i
                cum_ret_lstm_1 <- c(cum_ret_lstm_1, cumret_i)
  }
}

df_btc2$cumretlstm_1 <- cum_ret_lstm_1

cum_ret_lstm_2 <- c(0)

for (i in 2:nrow(df_btc2))  {
        if(df_btc2$change_pos_l2[i -1 ] == 1) {
                ret_i <- (df_btc2$btc_return[i]-0.0005)*df_btc2$pos_ls
tm_2[i - 1]
                cumret_i <- cum_ret_lstm_2[i - 1] + ret_i
                cum_ret_lstm_2 <- c(cum_ret_lstm_2, cumret_i)
  }     else {
                ret_i <- df_btc2$btc_return[i]*df_btc2$pos_lstm_2[i -
1]
                cumret_i <- cum_ret_lstm_2[i - 1] + ret_i
                cum_ret_lstm_2 <- c(cum_ret_lstm_2, cumret_i)
  }
}

df_btc2$cumretlstm_2 <- cum_ret_lstm_2

df_btc2$cumret_buyhold[2] <- df_btc2$cumret_buyhold[2] - 0.0005
df_btc2$cumret_buyhold[167] <- df_btc2$cumret_buyhold[167] - 0.0005

cum_ret_lstm_3 <- c(0)

for (i in 2:nrow(df_btc2))  {
        if(df_btc2$change_pos_l3[i - 1 ] == 1) {
                ret_i <- (df_btc2$btc_return[i]-0.0005)*df_btc2$pos_ls
tm3[i - 1]
                cumret_i <- cum_ret_lstm_3[i - 1] + ret_i
                cum_ret_lstm_3 <- c(cum_ret_lstm_3, cumret_i)
  }     else {
                ret_i <- df_btc2$btc_return[i]*df_btc2$pos_lstm3[i - 1
]
                cumret_i <- cum_ret_lstm_3[i - 1] + ret_i
                cum_ret_lstm_3 <- c(cum_ret_lstm_3, cumret_i)
  }
}

df_btc2$cumretlstm_3 <- cum_ret_lstm_3

x <- c(df_btc2$date, df_btc2$date, df_btc2$date, df_btc2$date, df_btc2
$date)
```

```
strategy <- c(rep("buy&hold", 167), rep("ARIMA+GARCH(1,1)", 167), rep(
"lstm_1", 167), rep("lstm_2", 167), rep("lstm_3", 167))
value <- c(df_btc2$cumret_buyhold, df_btc2$cumretAG, df_btc2$cumretlst
m_1, df_btc2$cumretlstm_2, df_btc2$cumret_lstm3)

btc_df_plot <- data.frame(x, strategy, value)

library(ggplot2)

ggplot(btc_df_plot, aes(x = x, y = value, group = strategy, colour = s
trategy)) +
  geom_line() +
  xlab("time") +
  ylab("Cumulative Return of each strategy") +
  ggtitle("Visualizing strategies performance for BTC")
```



We only show it with Bitcoin in this graph, because it is clear that the commission fees are almost negligible, given that the cumulative return and profit are so little affected. However, there are other fees we should consider, as the ones for transforming USD to BTC at the beginning (in Binance you cannot do that). These fees should be taken into account at the beginning. Nevertheless, their effect is equal for all strategies including Buy & Hold, therefore it does not make much sense to plot it with that detail, as the results

are very clear: ARIMA + GARCH(1, 1) defeats Buy & Hold even when having much higher trading costs, and the difference would still hold if the conversion fees were added.

ANNEX 6

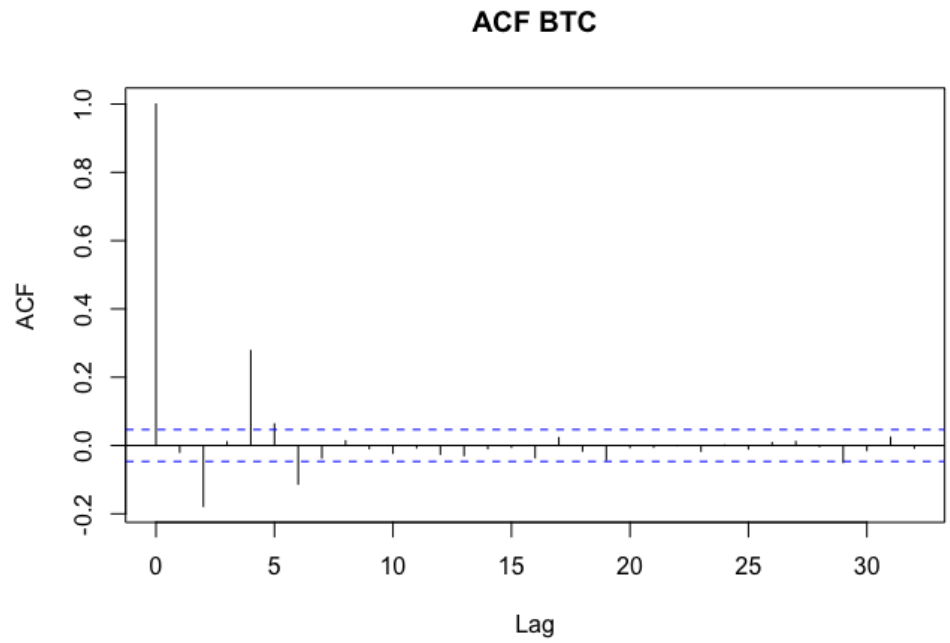


Figure 10. ACF bitcoin log returns.

Data Source: Cryptocompare (2018)

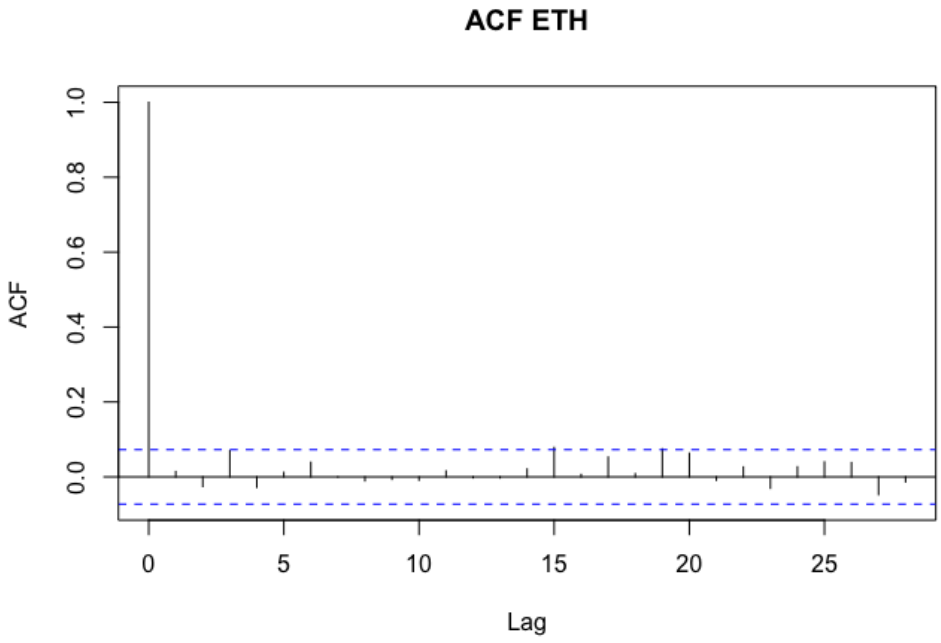


Figure 11. ACF eth log returns.

Data Source: Cryptocompare (2018)

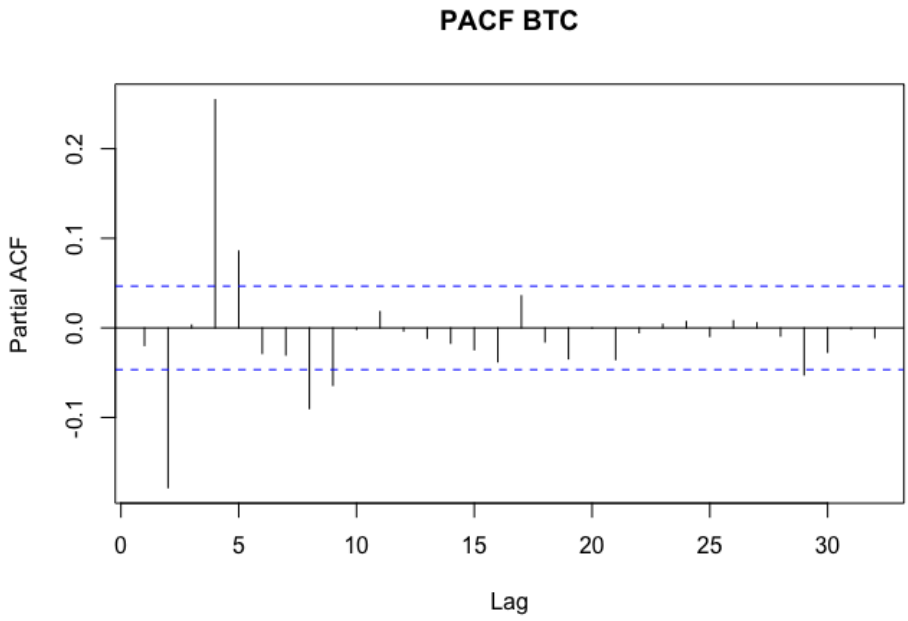


Figure 12. PACF btc log returns.

Data Source: Cryptocompare (2018)

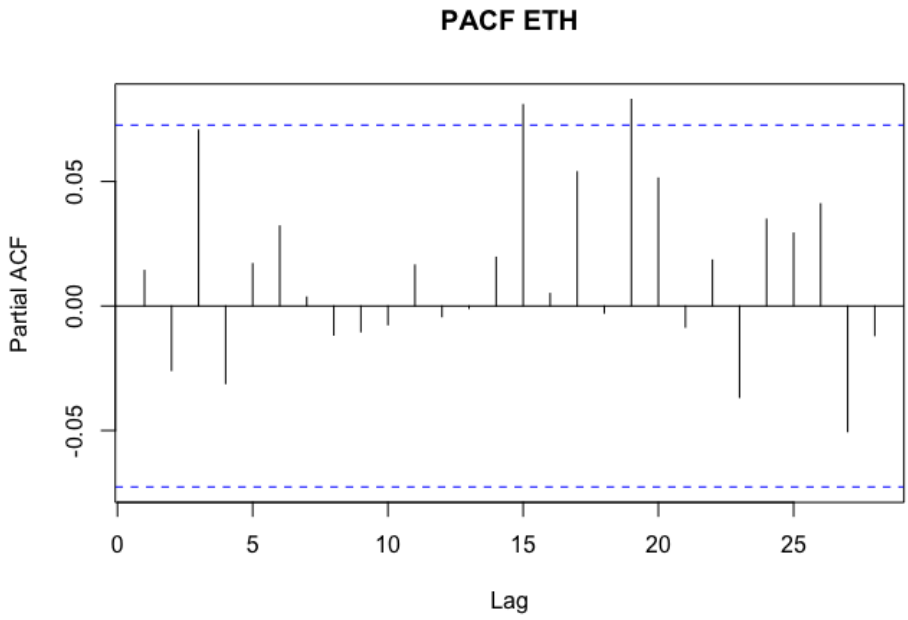


Figure 13. PACF eth log returns.

Data Source: Cryptocompare (2018)

Crypto	btc	eth
Ryan (2017) Overall Grade	9,2	9,6
Community Support	8,6	6,9
Developer Activity	9,8	9,4
Liquidity	10	9,2
tot_perc_ret	2,340743	2,655352
mean_daily_ret	0,006105703	0,006839364
av_google_interest	0,2456863	0,03607843
av_google_interest*10	2,456863	0,3607843
mean_daily_vol	863,044	59,27429
mean_daily_perc_vol	0,1028527	0,1102579
1/mean_daily_perc_vol	9,722642186	9,069644896
1/time	0,1316264	0,3941685
(1/time)*10	1,316264	3,941685
Mean	6,679564023	6,390933275
%of each in the group	51,10%	48,90%
% portfolio	**25,55%**	**24,45%**

Table 5. Core Cryptocurrencies' Portfolio Weights Calculation.

Data Sources: Cryptocompare (2018), Ryan (2017), Coingecko (2018), Google Trends (2018)

Crypto	xem	xrp	neo
Ryan (2017) Overall Grade	8	8,2	8
Community Support (%)	5,4	7,3	7,1
Developer Activity (%)	6,1	8,6	8
Liquidity (%)	6,5	8,9	7,4
tot_perc_ret	1,402948	5,437292	7,43707
mean_daily_ret	0,004787476	0,009371945	0,01150895
av_google_interest	0,3148077	0,1132692	0,7126923
av_google_interest*10	3,148077	1,132692	7,126923
mean_daily_vol	0,09538571	0,1119	10,43291
mean_perc_vol	0,1997265	0,185339	0,1842066
1/mean_daily_perc_vol	5,006846863	5,395518482	5,428687137
1/time	1,398467	0,3247331	1,852792
(1/time)*10	13,98467	3,247331	18,52792
Mean	6,192817733	6,026604185	8,627575017
%of each in the group	29,71%	28,91%	41,39%
% portfolio	**5,94%**	**5,78%**	**8,28%**

Table 6. Platform Cryptocurrencies' Portfolio Weights Calculation.

Data Sources: Cryptocompare (2018), Ryan (2017), Coingecko (2018), Google Trends (2018)

Crypto	xmr	zec	dash
Ryan (2017) Overall Grade	7,8	8,2	8
Community Support (%)	6,3	4,9	5,4
Developer Activity (%)	9	8,8	8,4
Liquidity (%)	7	7,1	7
tot_perc_ret	5,666319	1,33745	2,894846
mean_daily_ret	0,009607027	0,004395833	0,006850449
av_google_interest	0,05692308	0,03173077	0,7940385
av_google_interest*10	0,5692308	0,3173077	7,940385
mean_daily_vol	26,29138	45,33082	67,76372
mean_perc_vol	0,1414591	0,1334495	0,12294
1/mean_daily_perc_vol	7,069181127	7,493471313	8,13404913
1/time	0,3270609	0,7635983	0,2481305
(1/time)*10	3,270609	7,635983	2,481305
Mean	5,834417491	5,723026502	6,281323141
%of each in the group	32,71%	32,08%	35,21%
% portfolio	**6,54%**	**6,42%**	**7,04%**

Table 7. Anonimity Cryptocurrencies' Portfolio Weights Calculation.

Data Sources: Cryptocompare (2018), Ryan (2017), Coingecko (2018), Google Trends (2018)

Crypto	iota	lsk	maid
Ryan (2017) Overall Grade	8,75	na	8,6
Community Support (%)	6,7	6,4	4,3
Developer Activity (%)	7,1	8,7	7
Liquidity (%)	4	3,4	3
tot_perc_ret	4,094208	15,30806	0,2667234
mean_daily_ret	0,008573417	0,01424316	0,001079182
av_google_interest	0,09923077	0,01490385	0,005
av_google_interest*10	0,9923077	0,1490385	0,05
mean_daily_vol	0,3354153	2,117857	0,09970969
mean_perc_vol	0,2063972	0,1786537	0,1829106
1/mean_daily_perc_vol	4,845026967	5,597421156	5,467151712
1/time	1,46	1,01108	1,862245
(1/time)*10	14,6	10,1108	18,62245
Mean	6,385192833	7,095045665	5,913290639
%of each in the group	32,92%	36,58%	30,49%
% portfolio	**1,65%**	**1,83%**	**1,52%**

Table 8. Protocol Cryptocurrencies' Portfolio Weights Calculation.

Data Sources: Cryptocompare (2018), Ryan (2017), Coingecko (2018), Google Trends (2018)

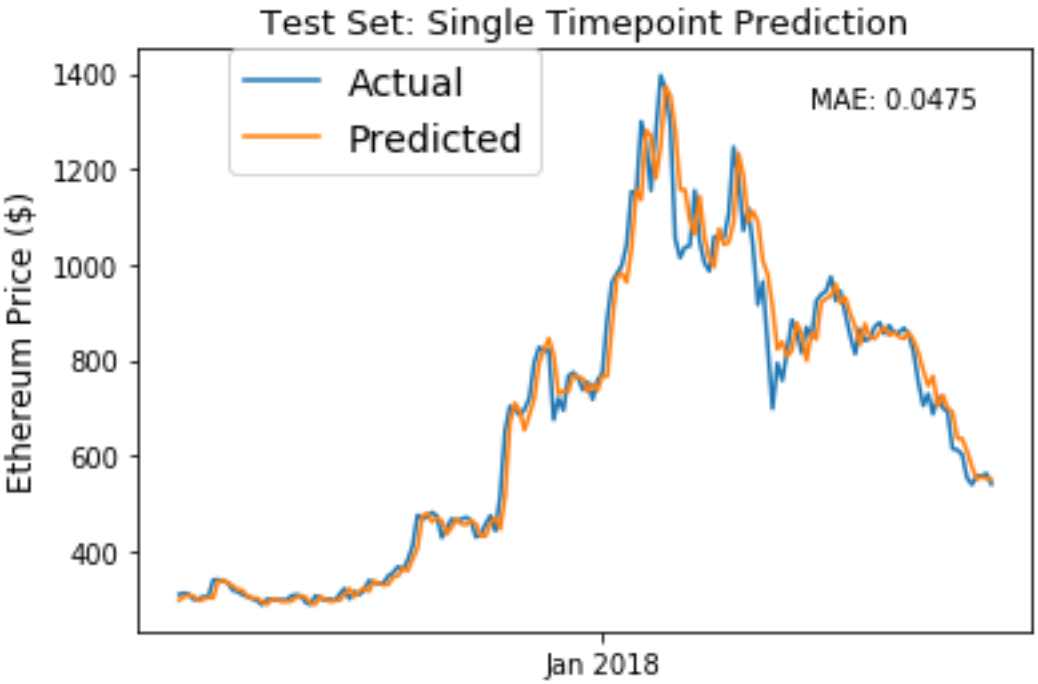


Figure 14. LSTM (1) ether predictions.

Data Sources: Coinmarketcap (2018) (1) (2) (3) (4)

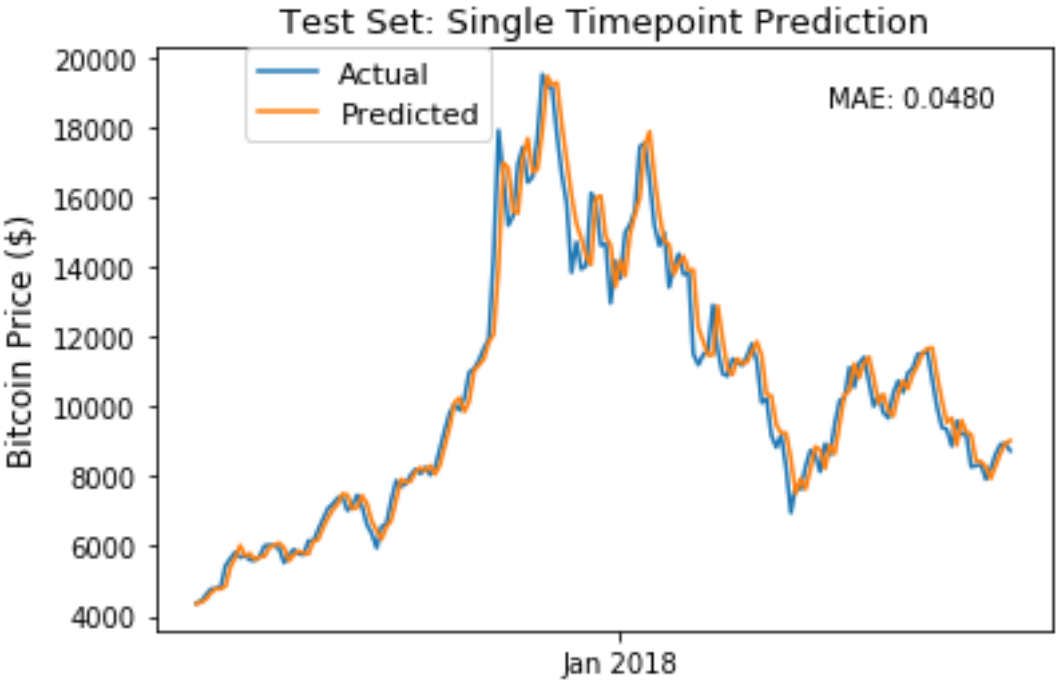


Figure 15. LSTM (1) bitcoin predictions.

Data Sources: Coinmarketcap (2018) (1) (2) (3) (4)

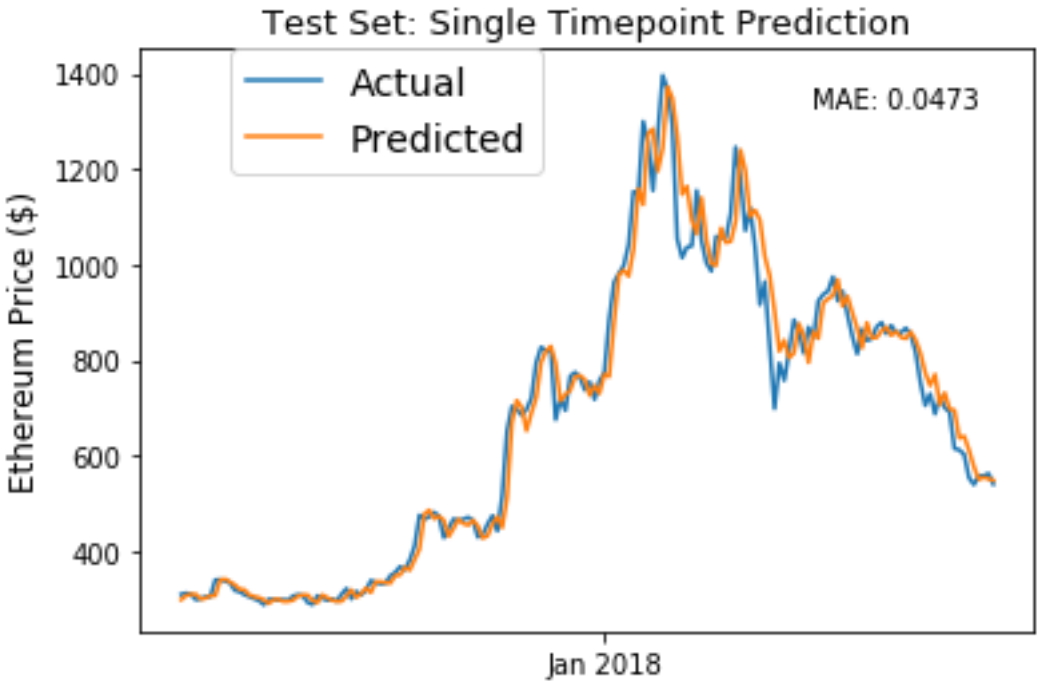


Figure 16. LSTM (2) ether predictions.

Data Sources: Coinmarketcap (2018) (1) (2) (3) (4)

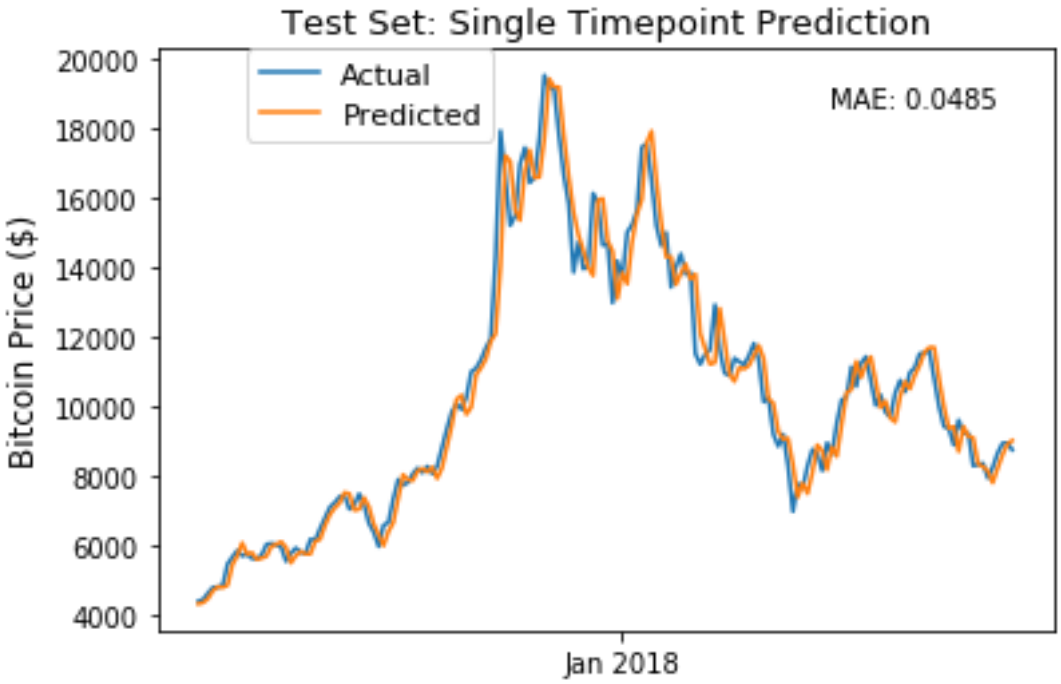


Figure 17. LSTM (2) bitcoin predictions.

Data Sources: Coinmarketcap (2018) (1) (2) (3) (4)

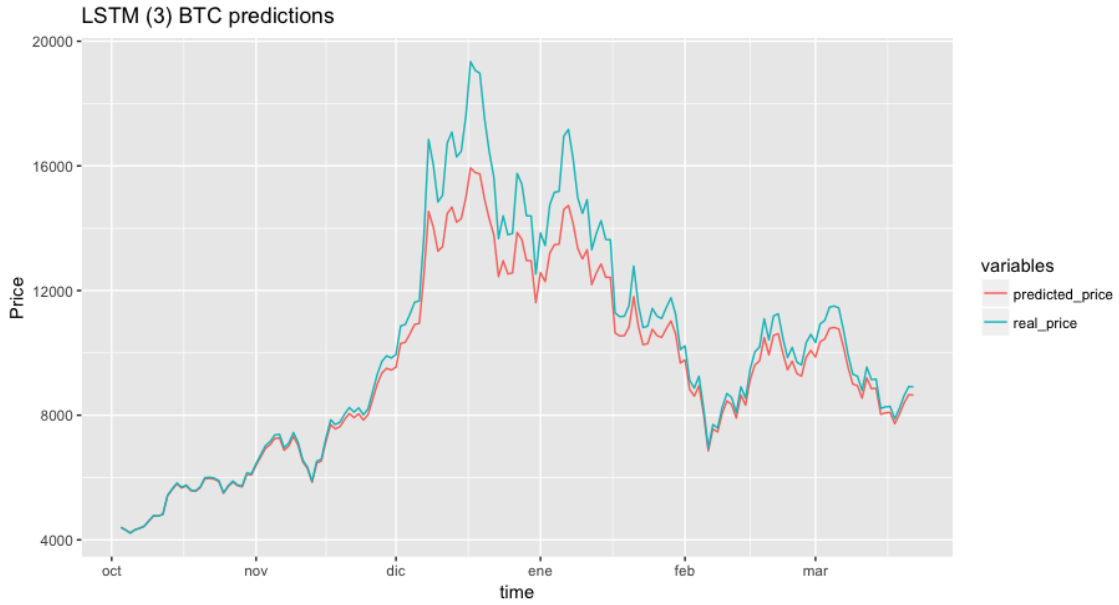


Figure 18. LSTM (3) bitcoin predictions.

Data Source: Cryptocompare (2018)

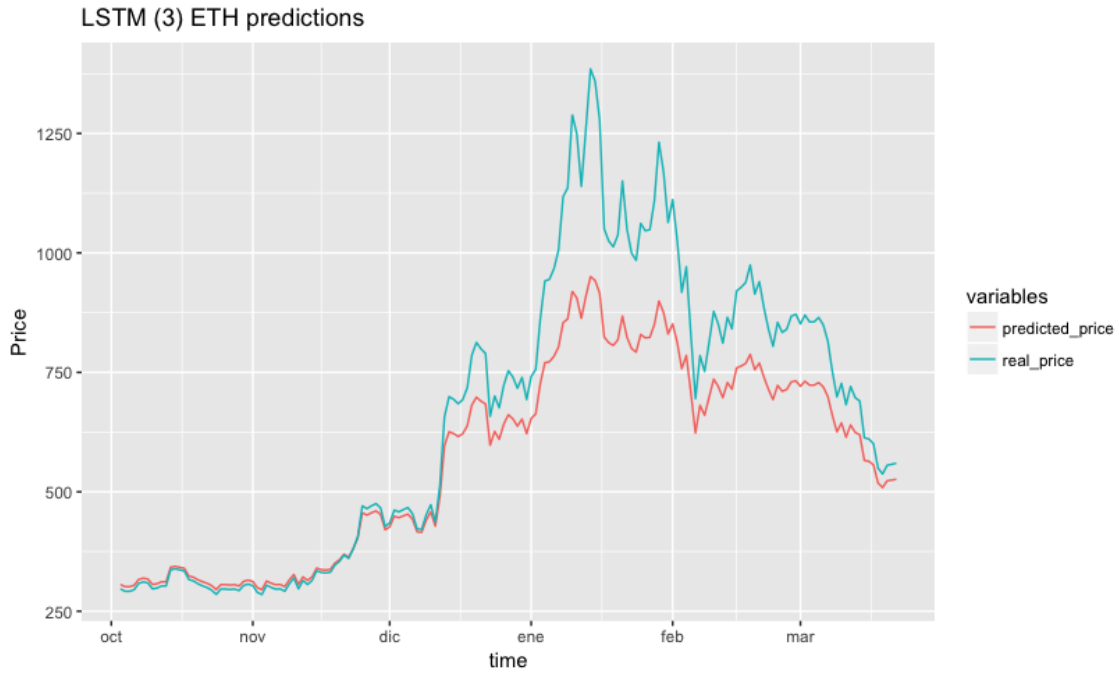


Figure 19. LSTM (3) ether predictions.

Data Source: Cryptocompare (2018)