POLITECNICO DI MILANO

Facoltà di Ingegneria dei Sistemi

Corso di Laurea in Ingegneria Gestionale

HIGH FREQUENCY TIME SERIES FORECASTING WITH AN APPLICATION TO STOCK MARKET PREDICTION

Relatore: Prof. Carlo VERCELLIS

Tesi di Laurea di – Master's thesis of:

Arnaud TREBAOL - Matr. 734348

Anno Accademico 2009 – 2010

POLITECNICO DI MILANO

Facoltà di Ingegneria dei Sistemi

Corso di Laurea in Ingegneria Gestionale

HIGH FREQUENCY TIME SERIES FORECASTING WITH AN APPLICATION TO STOCK MARKET PREDICTION

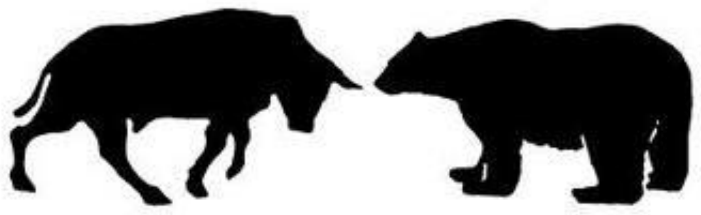
Relatore: Prof. Carlo VERCELLIS

Tesi di Laurea di – Master's thesis of:

Arnaud TREBAOL - Matr. 734348

Anno Accademico 2009 – 2010

A ma famille. Cette thèse est grâce à vous, elle est pour vous, elle est à vous.



ABSTRACT

Forecasting a stock price has been regarded as one of the most challenging applications of modern time series forecasting for many reasons: to yield significant profits for stock exchange investments on the one hand, and to refute the Efficient Market Hypothesis on the other hand. This work has for goal to compare different models for financial time series forecasting in the more specific field of high frequency data taken from the intraday French stock market “CAC 40”. Ten various companies stocks were used to match the prediction performances of linear time series models, non-linear time series models and the neural networks as a machine learning.

Keywords: stock price forecasting, high frequency time series, ARIMA, ARFIMA, exponential smoothing, STAR, ARMA-GARCH, artificial neural networks, intraday data, one-step-ahead prediction, forecasting performances, non-linearity.

SOMMARIO

Prevedere il valore di un titolo azionario è diventato una delle più grandi applicazioni nell'ambito della previsione delle serie storiche moderne per numerose ragioni: ottenere importanti profitti in primo luogo e rifiutare l'ipotesi dell'efficienza del mercato finanziario in secondo luogo. Questa tesi di laurea ha per obiettivo il confronto di diversi modelli per la previsione di serie storiche finanziarie e in particolare nell'ambito di dati ad alta frequenza relativi al mercato francese "CAC 40". Dieci diversi titoli di aziende sono stati usati per confrontare le previsioni di modelli lineari, di modelli non lineari e di reti neurali usati come apprendimento automatico supervisionato, valutandone le performance.

Parole chiave: previsione di titoli azionari, dati ad alta frequenza, ARIMA, ARFIMA, exponential smoothing, STAR, ARMA-GARCH, reti neurali artificiali, dati intraday, previsione un passo avanti, performance di previsioni, non linearità.

AKNOWLEDGMENTS

Vorrei ringraziare in primo luogo il Professore Carlo Vercellis del Politecnico di Milano per avermi dato la Sua fiducia, proponendomi un soggetto di tesi gratificante e in linea con le mie aspettative professionali, per avermi anche seguito durante tutto questo lavoro impegnativo con i suoi consigli e commenti, per fare di questo lavoro di laurea un vantaggio competitivo per il mio curriculum.

Je vais remercier en second lieu ma famille, mes parents et ma soeur, qui m'ont soutenu psychologiquement, qui m'ont donné la motivation nécessaire pour réussir ce travail et qui m'ont permis d'arriver là où j'en suis actuellement, aussi bien dans la vie étudiante et professionelle que personnelle. Sans eux, ce travail n'aurait pas été rondement mené, ils ont apporté une énorme pierre à l'édifice, une pierre précieuse, unique. La pierre précieuse c'est vous. Et Merci à ma tante Catherine pour son écoute et ses impressions-éclairs!

Je ne pourrais pas oublier l'aide, ô combien importante, de Maxime Juramy, étudiant en ingénierie mathématique au Politecnico di Milano, et à ses heures, un colocataire inoubliable, un ami. Ces connaissances en mathématiques et statistiques m'ont permis de progresser rapidement dans ma thèse, il n'a jamais feint à répondre à mes questions, parfois des plus ardues, comme son vier. Il m'a permis de gagner un temps précieux, temps qui est très cher dans une thèse de fin d'études. Avec cette thèse terminée je peux enfin dire que je suis fier come un Roux-casse!

Un grande ringraziamento a Lorenzo Vayno, detto Lolo, laureando in ingegneria matematica al Politecnico, per avermi fatto il piacere di essere il suo ospito durante la fine della tesi. Non avevo più nessun alloggio a Milano e mi ha permesso di finire tranquillamente la scrittura della tesi rimanendo vicino al Politecnico.

Francesco Trovò, ancora uno studente del Politecnico di Milano, specializzato in computer science: lo ringrazio molto di avere condiviso le sue conoscenze in neural networks e per avermi fatto il piacere di rispondere alle mie domande su quest'ambito.

I am really thankful for Joshua Ulrich and its CRAN package TTR dedicated to technical analysis and for our little but flourish mail communications; and for the Wiley company employees to have allowed me to get an access to diverse research articles mostly presented in the Journal of Forecasting, to help me to know the procedure to publish my work on this journal and also for their great reactivity.

I did not forget my friends, of whatever nationality they are, they brilliantly took their responsibilities in the good development of this work by their multiple encouragements and their communicative cheerfulness: Ania, Anna, Antonio, Benoit, Brice, Flavie, Isabelle, Jorge el wey, Jorge el español, Julien, Manuel, Nicolas, Pivounet, Sara, Stéphane, Valentina, Oliver Mouquin, "et al.".

Masha, tu as une grande place dans mon cœur, tu es mon avenir personnel et je te remercie d'être dans ma vie. Ma thèse est un aboutissement universitaire et tu m'as aidé dans la dernière ligne droite, la plus dure et la plus stressante; mais toi tu seras mon bonheur abouti, j'espère qu'avec toi nous bâtirons et nous aboutirons à de grandes choses. Je t'aime.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Time series are a special form of data where past values in the series may influence future values, depending on the presence of underlying deterministic forces. These forces may be characterised by trends, cycles and nonstationary behaviour in the time series and predictive models attempt to recognise the recurring patterns and more particularly potential linear or nonlinear relationships between past and actual values, or with other exogenous variables which may be linked to the variable studied.

Time series forecasting is the use of a model to forecast future time series values based on known past events: to predict data points before they are measured. Forecasting is an important and recurrent issue in business world since good forecasting models can lead to a major position in the market. Indeed a firm can anticipate the temporal evolution of a given data in order to implement solutions before its competitors. Forecasting problems find their applications in many fields: for example sales in marketing, production volume in operations and logistics, economic variable like GDP in macroeconomic studies or financial variables like stock prices in the finance field and more particularly in worldwide exchange.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on a financial exchange like also tax of interest. Stock movement prediction has been at focus for years since it can potentially yield significant profits by intelligent investments: we can mention for this the work of Yao et al. *Neural networks for technical analysis: a study on KLCI* in 1998 [32].

There are two approaches of investing in stock exchange. Some investors are interested in long terms and they look at the prospected future value of a firm. The others, many market actors like traders and market makers, use the market to mostly invest in the short term to also benefit from market inefficiencies or to use the possible goodness of forecasters as a finder of hidden information, in simple words to exploit trading opportunities. Evidently “short term” is subjective and one can be interested in daily stock value or in the highest frequency with tick by tick data. Data tick refers to any market data which shows the price and volume of every print. Studying and forecasting time series can be done for daily, monthly values or intraday values like for every minute and in this case we use the term high frequency data and high frequency time series since the time between two values is very short. Intraday stock market becomes interesting fields for many researchers; we can cite the recent works of Mills in 2001, *Statistical analysis of high frequency data from the Athens stock exchange* and in 2010 with Wang et al. *Nonlinearity and intraday efficiency tests on energy futures markets* [24]. These time series studied are thus called **high frequency financial time series.**

The successful prediction of a stock's future price could yield significant profit. Some believe that stock price movements are governed by the random walk hypothesis and thus are unpredictable. Others disagree and those with this viewpoint possess a myriad of methods and technologies which purportedly allow them to gain future price information. And this conflict can be easily linked with the Efficient Market Hypothesis, a field of so much interest and which is still discussed now by different analysts and researchers with different points of view like we can see in the work in 2004 of Timmermann et al. *Efficient market hypothesis and forecasting* [12]. Indeed if a forecaster can effectively forecast a stock market, the veracity of the efficiency of the market in question could be altered.

Fundamental and technical analyses describe the two different methods for the two different “term-concerned” approaches we told above. They were the first two methods used to forecast stock prices. The first one is commonly long-term-oriented; at the contrary the second one is mostly short-term-oriented:

Fundamental analysis is a method of evaluating a security that entails attempting to measure its intrinsic value by examining related economic, financial and other qualitative and quantitative factors. Fundamental analysts attempt to study everything that can affect the security's value, including macroeconomic factors (like the overall economy and industry conditions) and company-specific factors (like financial conditions and management). The end goal of performing fundamental analysis is to produce a value that an investor can compare with the security's current price, with the aim of figuring out what sort of position to take with that security (under-priced = buy the stock or take a long position, overpriced = sell the stock or take a short position).

On the other hand technical analysis is a method of evaluating securities by analysing statistics generated by market activity, such as past prices, transactions volume, high and low prices of a given period: which means studying the market activity. Technical analysts do not attempt to measure a security's intrinsic value, but instead use charts and other tools like indicators and signals to identify patterns that can suggest future activity. Technical analysts believe that the historical performance of stocks and markets are indications of future performance.

In a shopping mall, a fundamental analyst would go to each store, study the product that was being sold, and then decide whether to buy it or not. By contrast, a technical analyst would sit on a bench in the mall and watch people go into the stores. Disregarding the intrinsic value of the products in the store, the technical analyst's decision would be based on the patterns or activity of people going into each store.

A new method appeared more recently, the technological analysis, where computers are used as a tool to predict the stock movements. Technological analysis tries to model and simulate as accurately as possible the behaviour of the stock prices by different techniques; some of them can be more appropriate to study financial time series and to forecast them and more particularly to study high frequency financial time series.

In this thesis we will discuss different time series models to fit and forecast our different stock prices and we will compare them according to their forecasting performances. We will also try to conclude what could be the best one to yield potential significant profits. Our stock prices will be taken from high frequency data, so we will focus on the short term predictability of these different models and their possible applications on high frequency trading which means keeping buying and selling different stocks. We will discuss the models with a logical and progressive view by explaining the possible benefits from one model to another. Each model takes into account the possible importance of past values, so financial time series forecasting is linked to the technical analysis hypothesis. We will increase the degree of model complexity along the diverse chapters, beginning with typical time series models to evolve to machine learning which has a greater predictability power according to some works like Perez-Rodriguez et al. *STAR and ANN models: forecasting performance on the Spanish ”Ibex-35” stock index* [25] in 2005. This thesis will also put emphasis on the importance of linearity or nonlinearity in financial time series, which may show the suitable characteristic of non-linear models. All the chapters of this work will retake the different fields to be explored that we have introduced since now:

In **Chapter I**, we will expose some hypotheses, issues and problems about stock market forecasting and we will present an overview of financial modelling: a state of art about financial time series forecasting. Then the **Chapter II** will be dedicated to the overall description, to the statistical behaviour, to the different tests of our high frequency financial data and to the possible presence of non-linearity. The next logical step will be discussed in **Chapter III** with a presentation of our forecasting methodologies. **Chapter IV** will contain the theory about linear models, the first way used to forecast. **Chapter V** will take into account another type of models: non-linear models, which have to be concerned here to improve perhaps forecasting. **Chapter VI** will describe a more evolved model: machine learning and more particularly neural networks. Last but not least, we will finish this thesis and this final graduating work by the empirical results and their analysis in **Chapter VII**.

Warren Buffet, third worldwide capital property, who has beaten during many years the Dow Jones, would not introduce this work better than anyone:

“If markets were efficient, I would be a beggar”

HIGH FREQUENCY FINANCIAL FORECASTING

Stock price forecasting can be linked with the technical analysis since their goals are the same. Technical analysis is more a chart pattern recognition than mathematical models with detecting signals of buying or selling based on past experiences and some hypotheses that “technicians” assume:

- Market action discounts everything: based on the premise that all relevant information is already reflected by prices, pure technical analysts believe it is redundant to do fundamental analysis.
- Prices move in trend: technical analysts believe that prices trend directionally, i.e., up, down, or sideways (flat) or some combination.
- History tends to repeat itself: technical analysts believe that investors collectively repeat the behaviour of the investors that preceded them and with that the market psychology enters in action.

These hypotheses should not be so far from the ones of a mathematical modelling forecaster.

I. About stock price forecasting

Like we said before in the global introduction, financial time series and stock price time series are not a usual and typical time series to study. Indeed the predictability power is intrinsically linked to the Efficient Market Hypothesis. This means to understand the behaviour of stock market: do stock prices evolve as random walk and thus unpredictable? Or are they correlated to one or more variables, endogenous or exogenous? The answer can revolutionize the stock market understanding. That is why anyone cannot try to model and forecast financial time series without discussing about this intrinsic issue. But before discussing it we can describe the fluctuations mechanisms of stock price.

1. Stock price fluctuations

The price of a stock fluctuates fundamentally due to the theory of supply and demand. Like all commodities in the market, the price of a stock is sensitive to demand. It simply means that when a stock is demanded more than offered the price increases, at the contrary when it tends to be sold more than demanded the price decreases. If anyone car predict the future trend, the next trend of a stock, up or down, then this speculator or we can say predictor could obtain profit by buying it before it increases and selling it after, or selling it before it decreases and buying it after. This is the dynamics of offer and demand and with the term "dynamics" we can imagine non-linear behaviour. Mostly this dynamic has effects on a short-term viewpoint.

Furthermore, there are many factors that influence the demand for a particular stock: the good intrinsic results of the company, the profitable perspective results, the selling of a non-profitable business unit or not linked with the core business, share buy-back by the society or its managers and when the company is targeted by a tender offer; we can also mention macroeconomic factors like the favourable economic climate, company's sector perspectives, the decrease of interest rate, macroeconomic variables like inflation or unemployment, the rating credit of a rating agency, the exchange rate between two currencies, the gas or oil prices. Most of the time, these factors have a long term effect.

The stock price dynamic mechanism is essentially due to the offer and demand dynamics and it points out the field of market microstructure: this is a branch of finance concerned with the details of how exchange occurs in markets. While the theory of market microstructure applies to the exchange of real or financial assets, more evidence is available on the microstructure of financial markets due to the availability of transactions data from them. The major thrust of market microstructure research examines the ways in which the working processes of a market affects determinants of transaction costs, prices, quotes, volume, and trading behaviour. We can mention for this the work of Madhavan in 2000 *Market microstructure: a survey* [38] related to price formation.

2. Behavioural finance

There is another important field explored to understand the markets dynamics - and maybe inefficiencies - and more particularly the behaviour of market actors: the behavioural finance. The central issue in behavioural finance is explaining why market participants make systematic errors. Such errors affect prices and returns, creating market inefficiencies. It also investigates how other participants arbitrage such market inefficiencies. Behavioural finance highlights inefficiencies such as under - or over - reactions to information as causes of market trends and in extreme cases of bubbles and crashes. Such reactions have been attributed to limited investor attention, overconfidence, over-optimism, mimicry (herding instinct) and noise trading. Technical analysts consider behavioural economics' academic cousin, behavioural finance, to be the theoretical basis for technical analysis. A paper from Stracca in 2002 *Behavioural finance and asset prices: Where do we stand?*[39] contains a survey of the anomalies identified in the behavioural finance literature, with a particular focus on those which might affect market prices, like mimetic behaviour and emotional decisions; a large part of psychology is linked with market finance.

3. The Efficient Market Hypothesis (EMH)

Critics of the behavioural finance such as *Eugene Fama* typically support the efficient market hypothesis. In the 1970s *Eugene Fama* defined an efficient financial market as “one in which prices always fully reflect available information” and his PhD *"Random Walks In Stock Market Prices"* concluded that stock prices are unpredictable.

The EMH is an investment theory that states it is impossible to "beat the market" because stock market efficiency causes existing share prices to always incorporate and reflect all relevant information. In other words, the actual stock price instantly includes all information given to the market and we cannot find a useful factor to predict what the next value could be, so the most accurate prediction is the actual value. According to the EMH, stocks always trade at their fair value on stock exchanges, making it impossible for investors to either purchase undervalued stocks or sell stocks for inflated prices. As such, it should be impossible to outperform the overall market through expert stock selection or market timing, and the only way an investor could possibly obtain higher returns is by purchasing riskier investments and so not-surely-profitable. And it takes into account the impossibility of stock markets’ predictable capacities. Since information is taken by stock price value, no information could be useful to predict the next stock price. So the best prediction of the next value is the actual value because even if a price could be predictable, this information should be directly incorporated in the stock price without possibility to make profit.

Although it is a cornerstone of modern financial theory, the EMH is highly controversial and often disputed. Believers argue it is pointless to search for undervalued stocks or to try to predict trends in the market through either fundamental or technical analysis. The others like Professors Andrew W. Lo and Archie Craig MacKinlay, professors of Finance at the MIT Sloan School of Management and the University of Pennsylvania, respectively, have also tried to prove that the random walk theory is wrong. They wrote the book *A Non-Random Walk Down Wall Street*, which goes through a number of tests and studies that try to prove there are trends in the stock market and that they are somewhat predictable. Meanwhile, while academics point to a large body of evidence in support of EMH, an **equal** amount of dissension also exists. For example, investors, such as Warren Buffett have consistently beaten the market over long periods of time, which by definition is impossible according to the EMH. Detractors of the EMH also point to events, such as the 1987 stock market crash when the Dow Jones Industrial Average (DJIA) fell by over 20% in a single day, as evidence that stock prices can seriously deviate from their fair values and overall even in the short term. Even financial crises are mentioned arguments against the EMH.

We will also see in the following section of this chapter (I.5) some works and papers done in the field of research regarding stock price forecasting that could - or not - contradict the EMH.

There are actually different forms of market efficiency defined by Fama:

- In **weak-form efficiency**, future prices cannot be predicted by analysing prices from the past.
- In **semi-strong-form efficiency**, it is implied that share prices adjust to publicly available new information very rapidly and in an unbiased

fashion, such that no excess returns can be earned by trading on that information.

- In **strong-form efficiency**, share prices reflect all information, public and private, and no one can earn excess returns.

In our case we will try to model and forecast stock prices using past values for a short-term point of view so this is the weak form efficiency that is to be considered. If the forecasting methods are accurate and exploitable then the weak-form efficiency could be in contradiction and thus may be false.

In weak-form efficiency excess returns cannot be earned in the long run by using investment strategies based on historical share prices or other historical data. Technical analysis techniques will not be able to consistently produce excess returns, though some forms of fundamental analysis may still provide excess returns. Share prices exhibit no serial dependencies, meaning that there are no "patterns" to asset prices. This implies that future price movements are determined entirely by information not contained in the price series. Hence, prices must follow a random walk.

4. The benchmark: Random Walk theory

Another theory related to the efficient market hypothesis created by *Louis Bachelier* is the random walk theory, which states that the prices in the financial markets evolve randomly and are not connected to their past values: they are independent from each other. The random walk hypothesis is a financial theory stating that stock market prices evolve according to a random walk and thus the prices of the stock market cannot be predicted. This theory was also introduced by Fama. Therefore, identifying trends or patterns of price changes in a market could not be used to predict the future value of financial instruments. Thus the price would evolve like this:

$$X_t = \mu + X_{t-1} + \epsilon_t \tag{I.1}$$

Where X_t is the price of the stock at time t, μ is an arbitrary drift parameter, ε_t is a random disturbance term (white noise). The following figure shows the evolution of different random walks starting at time zero. This graphical view is quite similar to stock prices' graphs. We can see that in this case a stock price could not be predicted because his progress is totally random.

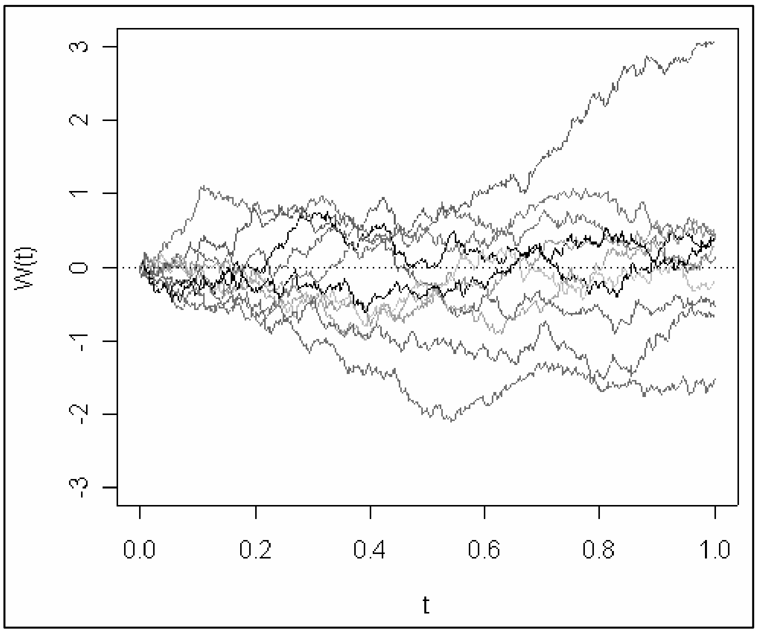


Figure 1: Random walks starting at the same point

So our goal is to beat the random walk, to find existing and recurrent patterns in past values in order to predict the next value. The random walk is thus our benchmark to compare our forecasting models according to adequate indicators we will exhibit later. The formula (I.1) also means that returns (difference between two successive stock prices) evolve like a white noise.

5. State of art: financial modelling and stock market forecasting

Many papers have been studied before writing this thesis as a documentation of what have been done in this field most recently and of the current way of research. We did in this part a literature overview regarding it. These works show that stock price forecasting have really been of great interest for many years and still nowadays. Different models to fit and predict stock market were used either to compare one model to another or to go against or to accept the weak form of the EMH. **These forecasting methods were applied in most cases to daily data. So high frequency forecasting is a current field of research, still in amelioration.**

The most part of these articles go against the random walk and the weak form of the EMH for stock market prices since they implemented models that could forecast stock prices better than random walk. The papers were mostly found on the website *directscience.com* taken from, also for the majority of cases, *The International Journal of Forecasting*. At the moment of the thesis writing, our work is studied to be published on this journal, I am working on it.

- In 2005 Pai et al. evaluated the usefulness of support vector machines (SVM) for a hybrid ARIMA-SVM model to forecast one-step-ahead stock price using ten daily stocks. This study demonstrated that a simple combination of the two best individual models does not necessarily produce the best results and is presented in *A hybrid ARIMA and support vector machines model in stock price forecasting* [3].

- Again in 2005, Bhardwaj et al. in *An empirical investigation of the usefulness of ARFIMA models for predicting macroeconomic and financial time series* [4] investigated the power of different models like ARMA, GARCH and ARFIMA to forecast daily returns at short, medium and long term. The conclusion was the outperformance of ARFIMA against random walk and the others models.
- More recently in 2009 Lu et al. in their paper *Financial time series forecasting using independent component analysis and support vector regression* [14] investigated the performance of the support vector regression using independent component analysis for the inputs as a forecaster of the daily NIKKEI 225. Experimental results showed that the proposed model can produce lower prediction error and higher prediction accuracy and outperformed the SVR and random walk models.
- Kim in 2003 also studied SVM in *Financial time series forecasting using support vector machines* [15]. This study used SVM to predict future direction of stock price index. In addition, this study compared SVM with Back Propagation Network. The experimental results showed that SVM outperformed BPN in the daily Korea composite stock price index (KOSPI).
- Tang et al. in 2003 in *Finite Mixture of ARMA-GARCH Model for Stock Price Prediction* [16] analyses the ARMA-GARCH to forecast HSBC stock price for a one-step-ahead prediction.
- Clements et al. in 2004 with their paper *Forecasting economic and financial time-series with non-linear models* [17] discuss the current state-of-the-art in estimating, evaluating, and selecting among non-linear forecasting models for economic and financial time series.
- Moreno et al. worked on the economical usefulness of stock market predictability studying the impact of cost transactions and comparing linear

models with non-linear models neural networks, in 2007 with *Is the predictability of emerging and developed stock markets really exploitable?* [19]. Their conclusions suggest that, in contrast to some previous studies, if we consider total trading costs both the emerging as well as the developed stock returns are clearly non-predictable. Finally, they find that Artificial Neural Networks do not provide superior performance than the linear models. They studied daily and weekly data for a one-step-ahead prediction.

- Thomakos et al. compared in their work subtitled *Naïve, ARIMA, nonparametric, transfer function and VAR models: A comparison of forecasting performance* [20] the forecasting performance of ARIMA, VAR models (multivariate models) and other models mentioned in the title of the paper. They find that the performance of these models is better than that of the naïve, no-change model. They used quarterly data for a one-step and multi-step ahead prediction.
- McMillan in 2007 with his paper *Non-linear forecasting of stock returns: Does volume help?* [22] compared, for financial daily data index, different models for a one-step-ahead prediction. He used a LSTAR model with the volume as the threshold variable to be compared with an AR and the random walk. This LSTAR outperformed the two last models. He emphasized the superiority of non-linear models against linear models. He also pointed out the importance of transaction costs for an economical usefulness.
- *Nonlinearities, cyclical behaviour and predictability in stock markets: international evidence* [23] by Sarantis in 2001 pointed out the non-linearity characteristic in stock index. Tests reject linearity for all stock markets. Their evidence on out-of-sample forecasting suggests that forecast gains can be made by exploiting the nonlinear structure of STAR models and that these models outperformed linear autoregressive models and random walks at both short term and medium term horizons.

- *Nonlinearity and intraday efficiency tests on energy futures markets* written by Wang et al. [24] tested the intraday market efficiency using high frequency data. The paper employs various nonlinear models and several model evaluation criteria to examine market efficiency in an out-of-sample forecasting context. Overall, there is evidence for intraday market inefficiency of two of the four energy future markets (heating oil and natural gas), which exists particularly during the bull market condition but not during the bear market condition. The data used consists of 30-minute intraday prices and returns. Using high frequency data, this paper first time comprehensively examines the intraday predictability of four major energy (crude oil, heating oil, gasoline, natural gas) futures markets. The paper employs various nonlinear models (neural network, semi-parametric functional coefficient model, nonparametric kernel regression, GARCH), in addition to a linear model, and several evaluation criteria based on both statistical and economic accuracy to examine market efficiency in an out-of-sample forecasting context. Overall, more thorough allowance for nonlinearity and market conditions (bear and bull markets) still only suggests somewhat limited evidence for intraday market inefficiency of energy future prices.
- In 2005, Pérez-Rodìguez et al. compared AR, STAR and Neural Networks models to forecast daily returns for the Ibex-35 index in *STAR and ANN models: forecasting performance on the Spanish Ibex-35Q stock index* [25]. This comparison was carried out on the basis of various statistical criteria and by assessing the economic value of the predictors. In both cases, they used different forecasting methods (one-step and multi-step-ahead predictions) and forecast horizons to analyse the robustness of the results. The results obtained put into question the hypothesis of Efficient Markets, because the random walk model is not a reasonable description of Ibex-35 stock index returns and the ANN technique may be an appropriate way to

improve forecasts. In the light of the results obtained, the statistical criteria suggest that the out-of-sample ANN forecasts are more accurate, they provide a better fit and provide slightly improved predictions than do the AR and STAR models.

- Atsalakis did a really great job in his recent paper in 2009: *Surveying stock market forecasting techniques – Part II: Soft computing methods* [27]. He reviewed a large number of papers (around one hundred) which were dedicated to implement model to forecast daily stock prices and were written from 1993 to 2006. These models were mostly different neural networks. He described the different data set used, the input variables for the networks, the different performance measures used and the different benchmarks like the random walk and the ARMA models. More specifications about neural networks were mentioned: sample size, transfer function, data pre-processing, number of hidden neurons and validation set. Given stock market model uncertainty, soft computing techniques are viable candidates to capture stock market nonlinear relations returning significant forecasting results with not necessarily prior knowledge of input data statistical distributions. Through the surveyed papers, it is shown that soft computing techniques are widely accepted to studying and evaluating stock market behaviour. The input variables mostly come from technical indicators, fundamental indicators and past values.
- Hassan et al. studied a non-typical model in 2007: *A fusion model of HMM, ANN and GA for stock market forecasting* [40] to forecast a one-step-ahead daily price. In this paper they described a novel time series forecasting tool. The fusion model combines a Hidden Markov Model (HMM), Artificial Neural Networks (ANN) and Genetic Algorithms (GA) to forecast financial market behaviour. As a result they find that the performance of the fusion tool is better than that of the basic model (Hassan & Nath, 2005) where only a single HMM is used in a novel approach to forecast stock price. To

evaluate the efficacy of the fusion model they compare the obtained forecast accuracy with that of a popular statistical forecasting tool (MAPE). The comparison shows the forecasting ability of the fusion model is as good as that of ARIMA model.

- Ellis et al. in 2004 and *Another look at the forecast performance of ARFIMA models* [41] studied the performance of ARFIMA models to forecast stock market prices. Using a variety of different measures of forecast performance, the ARFIMA specification fails to outperform forecasts derived from both the simple average and AR(1) models, and performs only as good as a forecast based on the last observed value or a random walk model.
- Chen et al. studied the *Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index* [42] for forecasting monthly data. The good performance of the Probalistic Neural Network suggests that the neural network models are useful in predicting the direction of index returns. Furthermore, PNN has demonstrated a stronger predictive power than both the GMM–Kalman filter and the random walk forecasting models.
- Hauser et al. worked on a hybrid ARFIMA-ARCH model to forecast high frequency financial time series: *Forecasting High-frequency Financial Data with the ARFIMA–ARCH model* [43].
- Mostafa studied the forecasting performance of neural networks on financial time series with *Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait* [44]. His results confirm the theoretical work by Hecht-Nielson (1989) who has shown that NNs can learn input–output relationships to the point of making perfect forecasts with the data on which the network is trained. The good performance of the NN models in predicting KSE closing price movements can be traced to its inherent non-linearity. This makes an NN ideal for dealing with non-linear

relations that may exist in the data. Thus, neuro-computational models are needed to better understand the inner dynamics of stock markets. His results are also in line with the findings of other researchers who have investigated the performance of NN compared to other traditional statistical techniques, such as regression analysis, discriminant analysis, and logistic regression analysis.

- Huang et al. worked on *Forecasting stock market movement direction with support vector machine* [45]: in this paper, they studied the use of support vector machines to predict financial movement direction. According to their work, SVM is a promising type of tool for financial forecasting. As demonstrated in their empirical analysis, SVM is superior to the other individual classification methods in forecasting weekly movement direction of NIKKEI 225 Index, like a neural network and the random walk.
- A neuro-fuzzy adaptive control system has been developed by Atsalakis et al. to forecast next day's stock price trends. The proposed system has been presented and described from conceptual and technical perspectives, justifying its modelling aspects. Obtained results challenge the weak form of the EMH, by demonstrating that when using historical data, accurate predictions of stock price trends are achievable. This statement has been supported by several case studies of different stocks from the ASE (an emerging market) and the NYSE (a well-developed market). The proposed system has performed very well in trading simulations, returning results superior to the Buy and Hold (B&H) strategy. This paper *Forecasting stock market short-term trends using a neuro-fuzzy based methodology* [46] used daily data and one-step-ahead predictions.
- Molgedey et al. studied the *Intraday patterns and local predictability of high-frequency financial time series* [47]. Their results show that local analysis is an appropriate tool for studying the predictability of financial time series. Of particular interest are local studies of the continuations and

predictabilities of certain local histories. Local correlations are of specific interest since they improve the local predictability. Hence, one can, in principle, improve the predictions at certain time instants by basing the predictions on local history observations. Further, they concluded that there are specific substrings which rarely occur and for which the uncertainty is noticeably less than 1; the local predictability is better than 10%. In other words, there are specific situations where the predictability is better than the average predictability. However, the effect is quite small and shows that the discussed financial time series is nearly random, but not fully random and shows some order at specific sub-trajectories.

- Strozzi et al. wrote a paper on *Non-linear forecasting in high-frequency financial time series* [48]. In a general way but applied to foreign exchange time series, a new methodology based on state space reconstruction techniques has been developed for trading in financial markets. The methodology has been tested using 18 high-frequency foreign exchange time series. The results are in apparent contradiction with the efficient market hypothesis which states that no profitable information about future movements can be obtained by studying the past prices series. In their analysis positive gain may be obtained in all those series.
- Sazuka used *Non-linear logit models for high-frequency data analysis* [49]. In this paper, they have analysed tick-by-tick data, the most high-frequency data available, of yen–dollar exchange market with focus on up down price movements. The proposed non-liner logit model has been able to capture the non-trivial probability structure in the binary tick-by-tick data, which was impossible using the conventional methods such as AR models or normal logit models. This model is a good model to forecast the probability of up or down trends conditionally to the up or down trends arrived before.
- *Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods* written by de Faria et al. [50] in 2009

compared two models the first one non-linear and the second one linear to forecast one-step-ahead stock markets. In this work the daily principal index of the Brazilian stock market was studied through artificial neural networks and also by using the adaptive exponential smoothing method. It is shown that the neural networks have a superior performance to predict the correct sign of the index return. The AES method did not contribute to predict the correct sign of the return, in spite that both methods, NN and AES, produced almost the same RMSE in the prediction of the return values.

- Resta in his 2009 paper *Seize the (intra)day: Features selection and rules extraction for tradings on high-frequency data* [51] showed that the results obtained are quite promising, since they give evidence that a trading system which combines rules and unsupervised neural networks can work well to discover intraday patterns, and to employ such retrieved information to perform market tradings.
- Enke et al. studied *The use of data mining and neural networks for forecasting stock market returns* in 2005, [52]. The results showed that the trading strategies guided by the neural network classification models generate higher profits under the same risk exposure than those suggested by the other strategies, including the buy-and-hold strategy, as well as the level estimation forecasts of neural network and linear regression models.

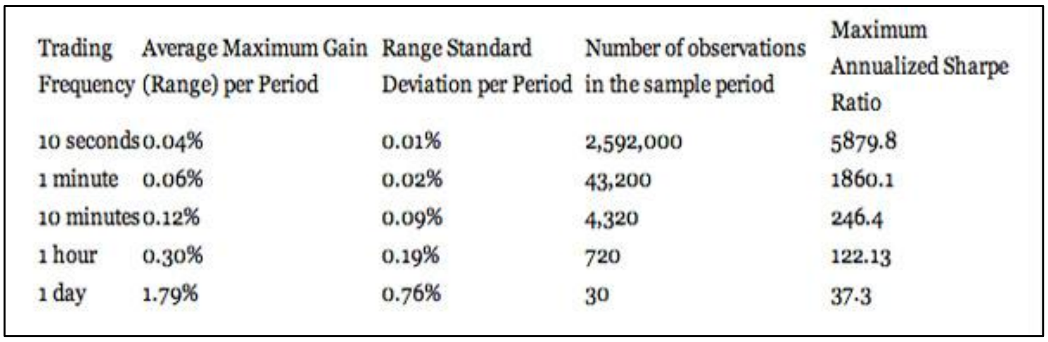
Much of these researches and also implicitly mine have attempted to answer two questions:

- ➢ Is the most accurate forecast of tomorrow’s price simply today’s price plus an estimate of the long-run average daily price change?
- ➢ Can profits be made by frequently changing a market position, buying and selling the same goods many times?

6. High frequency trading: the application

High-frequency trading is the execution of computerized trading strategies characterized by extremely short position-holding periods. In high-frequency trading, programs running on high-speed computers analyse market data, using algorithms to utilize trading opportunities that may open up for only a fraction of a second to several hours. High-frequency trading, often abbreviated HFT, uses quantitative investment computer programs to hold short-term positions in equities, options, futures, currencies, and all other financial instruments that possess electronic trading capability. High frequency traders compete on a basis of speed with other high frequency traders, not long term investors (who typically look for opportunities over a period of weeks, months, or years with , like we said before, fundamental analysis), and compete with each other for very small, and very consistent profits. As a result, high-frequency trading has been shown to have a potential Sharpe ratio thousands of times higher than the traditional buy-and-hold strategies [9]. A Sharpe ratio measures return per unit of risk; a Sharpe ratio of 2 means that the average annualized return on the strategy twice exceeds the annualized standard deviation of strategy returns: if the annualized return of a strategy is 12%, the standard deviation of returns is 6%. The Sharpe ratio further implies the distribution of returns: statistically, in 95% of cases, the annual returns are likely stay within 2 standard deviations from the average. In other words, in any given year, the strategy of Sharpe ratio of 2 and annualized return of 12% is expected to generate returns from 0% to 24% with 95% statistical confidence, or 95% of time. Table 1 compares the maximum Sharpe Ratios that could be attained at 10-second, 1-minute, 10-minute, 1-hour and 1-day frequencies in EUR/USD.

Table 1: Performance limits for trading strategies running at different frequencies



Trading Frequency	Average Maximum Gain (Range) per Period	Range Standard Deviation per Period	Number of observations in the sample period	Maximum Annualized Sharpe Ratio
10 seconds	0.04%	0.01%	2,592,000	5879.8
1 minute	0.06%	0.02%	43,200	1860.1
10 minutes	0.12%	0.09%	4,320	246.4
1 hour	0.30%	0.19%	720	122.13
1 day	1.79%	0.76%	30	37.3

As Table 1 shows, the theorical maximum profitability of trading strategies measured using Sharpe ratios increases with increases in trading frequencies. From March 11, 2009, through March 22, 2009, the maximum possible annualized Sharpe ratio for EUR/USD trading strategies with daily position rebalancing was 37.3, while EUR/USD trading strategies that held positions for 10 seconds could potentially score Sharpe ratios well over 5,000 (five thousand) mark.

The Sharp Ratio is defined as following:

$$S = \frac{R - R_f}{\sigma} = \frac{E[R - R_f]}{\sqrt{\mathrm{var}[R - R_f]}}, \qquad \textbf{(I.2)}$$

Where R is the asset return, R_f is the return on a benchmark asset, such as the risk free rate of return, $E[R - R_f]$ is the expected value of the excess of the asset return over the benchmark return, and σ is the standard deviation of the asset. This ratio has to be positive in order to conclude that the portfolio is more profitable than average.

II. High frequency financial data: description

Statistical finance, sometimes called econophysics, is an empirical attempt to shift finance from its normative roots to a positivist framework using exemplars from statistical physics with an emphasis on emergent or collective properties of financial markets. The starting points for this approach to the understanding of financial markets are the empirically observed stylized facts. In financial econometrics, a stylized fact is a structural observation that is believed to hold for a diverse collection of instruments, markets, and time periods. They are many stylised facts for financial time series that have been found in many research works, we can mention for this the paper of Cont in 2001 *Empirical properties for asset returns: stylised facts and statistical issues* [53]. Cont described some important facts about asset returns. Here are the most common which we can also find in [6]:

1. *Absence of autocorrelations*: (linear) autocorrelations of asset returns are often insignificant, except for very small intraday time scales (20 minutes) for which microstructure effects come into play. We can see this in figure 30 and the others from the appendix 1.
2. *Heavy tails* for the probability distribution like we can in figure 29 and the others from the appendix 1.
3. *Variance clustering*: variance often changes over time, with alternating phases of high and low volatility. Such behaviour is well captured by ARCH or GARCH models. Different measures of volatility display a positive autocorrelation over several days, which quantify the fact that high-volatility events tend to cluster in time. The white test can confirm it.
4. Financial time series are very complex and dynamic as they are characterized by extreme volatility.

5. Empirical evidence strongly suggest that the probability distribution functions found in financial time series exhibits a fat-tailed distribution which is in disagreement with the Gaussian distribution and the random walk model.
6. *Aggregational Gaussianity*: as one increases the time scale t over which returns are calculated, their distribution looks more and more like a normal distribution. In particular, the shape of the distribution is not the same at different time scales.
7. *Slow decay of autocorrelation in absolute returns* like we can see on the following figure. The ACF of the squared returns are quite identical.

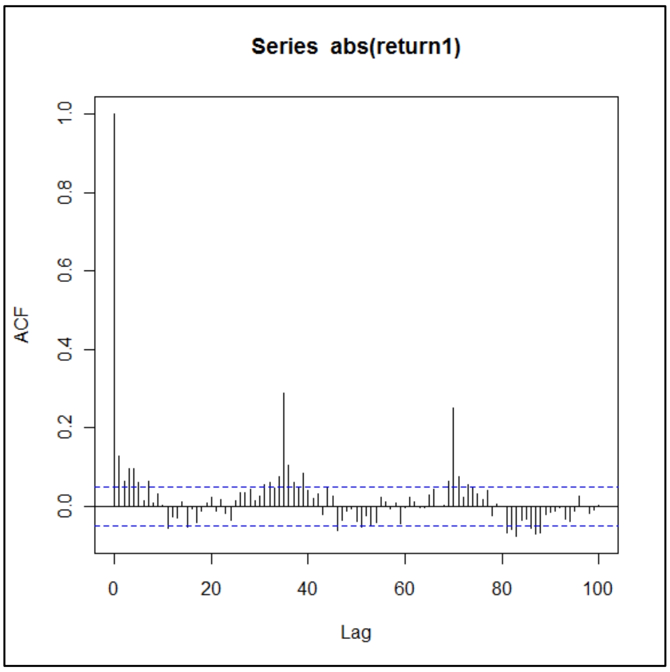


Figure 2: ACF of absolute returns

8. *Leverage effect*: most measures of volatility of an asset are negatively correlated with the returns of that asset. Confirmed by the use of R.

9. *Volume/Volatility correlation*: trading volume is correlated with all measures of volatility. Also confirmed by the use of R.

In this second section we will describe more particularly high-frequency financial data and our typical stocks used.

1. High frequency data

Stylised facts have also been exhibited for high-frequency financial data and we can see them on [5]:

- A significant negative autocorrelation within 4 minutes of trading is found in HF data, first reported by Goodhart and Figlioli (1991). They explain it with microstructure effects, as well as by formulating the hypothesis of diverging opinions of traders about the impact of news on the direction of prices.
- The estimate of the tail index indicates fat non-Gaussian tails of the distributions.
- Scaling properties indicate fractal behaviour of stock prices, and thus the possibility of extending HF findings to lower frequencies as well.
- Volatility autocorrelations decay at a hyperbolic (rather than exponential) rate. This indicates the presence of a ‘‘heat wave effect’’: We may not know when a stone falls in the lake, but we can predict its waves. Volatility also leaves a memory effect. This is not interpreted as a delay in price adjustment that signals inefficiency in reflecting information. Rather, the empirical findings indicate that volatility is the footprint of market presence, activity, and volume. We can understand with the following figure:

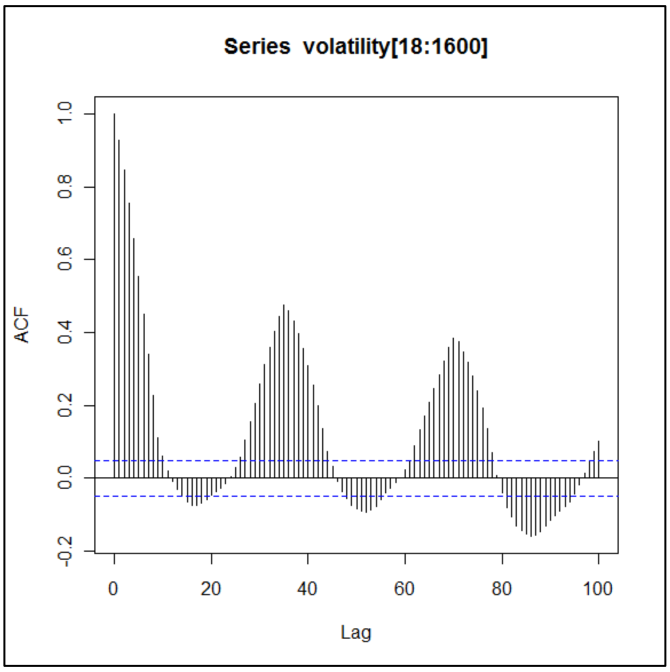


Figure 3: ACF for volatility

2. Our dataset

To do this study we chose ten diverse stocks taken from the French stock exchange CAC 40. The stocks were chosen in order to have companies working on diverse areas, in order to make the dataset as various as possible to be able generalize the results. The data were chosen as high frequency with time interval between two values equals to 15 minutes. Our dataset takes its values from the 09/10/2009 at 09:00 to the 11/12/2009 at 14:15, so two months and 1600 values.

We divided the dataset into a training set and a validation set: 1500 values for the training set and 100 values for the validation set (or test set).

For each time t of the period studied, the set of values is composed by an open value and a close value for the time at t+15 (normally, the close price for the period t: t+15 is equivalent to the open price of the period t+15: t+30). Between theses times (like t and t+15, and we call this the high frequency trading period), there are a high price and a low price of the period, corresponding to the highest and lowest price of the trading period in question. We have also the transaction volume of the period corresponding to the number of stocks that have been sold and bought.

Each company will be called with its number like for company 1 is [1] and company 2 is [2].

- The company 1 is, working on
- The company 2 is, working on
- The company 3 is, working on
- The company 4 is, working on
- The company 5 is, working on
- The company 6 is, working on
- The company 7 is, working on
- The company 8 is, working on
- The company 9 is, working on
- The company 10 is, working on

3. Basic statistics

First of all, before making any forecasting we have to statistically study our dataset with a statistical description and basic statistics. We can first of all see all the graphics from the **appendix 1**:

- The plot of the stocks to see how they evolve and if they are stationary (some of them can have a significant trend)
- The histogram of the stocks to see their distribution and that there is a significant asymmetry for mostly stocks.
- The boxplots of the stocks show that there are noteworthy outliers.
- The qq-plots of the stocks show a tendency to be normal but there are fat tails.

We can also see these statistics for the returns (difference between two consecutive stock values). We can see that there are more outliers for returns and less asymmetry for the distribution. The return distribution is narrower than the stock distribution. The returns seem to have a zero mean.

We can also describe the autocorrelation diagram and the partial autocorrelation diagram. For the stocks there is a slow decay for the ACF and for the returns not evident correlations can be found but only for the precedent value.

The graphical description is necessary to have a first view of the data studied. Then we can deepen on this way with statistical indicators. Here are the tables with the principal statistics indicators of each time series (for stocks):

Table 2: basic stats for [1] to [5]

Basic stats.	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**
Minimum	31.955000	12.650000	73.120000	46.500000	2.178000
Maximum	38.400000	15.620000	82.070000	52.790000	3.375000

1th quartile	34.848750	13.650000	77.480000	48.088750	2.342750
3rd quartile	37.091250	14.826250	79.392500	49.711250	3.005000
Mean	35.799853	14.227747	78.253163	49.057681	2.670885
Median	36.042500	14.340000	78.555000	48.900000	2.588000
Variance	2.442702	0.443587	3.005088	1.770254	0.116689
Stand. dev.	1.562914	0.666023	1.733519	1.330509	0.341597
Skewness	-0.588705	-0.219842	-0.632047	0.603640	0.575114
Kurtosis	-0.397300	-1.073876	0.272573	0.012136	-1.003767

Table 3: basic stats for [6] to [10]

Basic stats.	[6]	[7]	[8]	[9]	[10]
Minimum	39.555000	50.800000	29.045000	29.975000	15.600000
Maximum	44.020000	58.540000	33.840000	36.945000	19.351100
1th quartile	40.870000	54.320000	30.660000	30.998750	16.346200

3rd quartile	41.851250	56.655000	32.386250	33.663750	17.868350
Mean	41.376572	55.314450	31.438800	32.387728	17.091106
Median	41.220000	55.290000	31.380000	31.432500	16.748600
Variance	0.647039	2.994156	1.305531	3.781693	1.035828
Stand. dev.	0.804387	1.730363	1.142599	1.944658	1.017756
Skewness	0.872820	-0.318824	0.114416	1.010679	0.794798
Kurtosis	0.958059	-0.502973	-0.889888	-0.446801	-0.698521

We can first say that we can heterogeneous data since they have diverse mean, variance, Skewness and Kurtosis, so a different distribution. This is an essential criterion in order to assume general results. Like we can see on the skewness, each time series is asymmetric, either on the left or on the right. The kurtosis shows the gap with the normality distribution. Like we can see on the indicators, the distribution of the stock time series are not so much normal and more hypo-normal than hyper-normal (at the contrary like we can see on the statistical graphs for the returns, the returns tend to be very hyper-normal; the kurtosis is for each return superior to 8) with values not so high (inferior to abs(1)). We can see that the stock [4] has a normal distribution with a kurtosis value of 0.01.

We can also see the graph of correlation between the closing price and the volume with is typical and show non-linearity but two parts of linearity (low volume and high volume, with a discontinuity between the two parts)

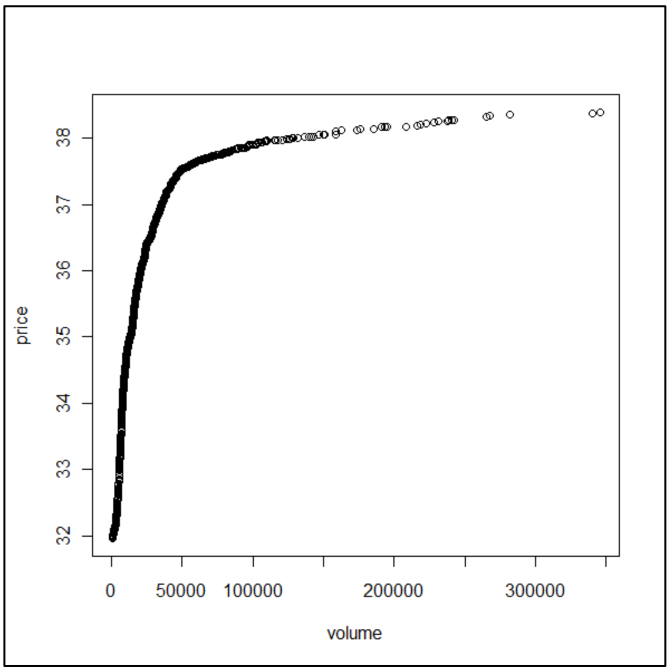


Figure 4: Correlation between stock and volume

4. Statistical Tests

We also did some tests with R to better understand our time series. These tests are concerning stationarity, normality and linearity. First we tried to find periodical parts in our time series and with the *decompose()* function on R, the result was no period part for each time series.

a. Stationarity

With tests of stationarity like the kpss test or the adf test (unit root test) we found that no stock is stationary and the return are at the contrary stationary.

b. Normality

We also tested the normality with the Shapiro test and this one confirmed that no stock or return is normal.

c. Linearity

We tested the linearity of our time series because like we can see on the previous researches, financial time series seem to be non-linear and non-linear models seem to better fit the data. With adequate tests (like White test or Teräsvirta test) we found that [1], [3], [7] and [10] are said to be non-linear, the others are found to be non-linear.

d. Independence

Tests for independence and identically distributed (i.i.d.) time series for the stocks and the returns show that that the i.i.d. was wrong. (see bds test for example)

III. Forecasting methodologies

We will present in this chapter the different forecasting methodologies we used to do this work. This is our bases of study. To remind to the reader, our goal is to compare different models forecasting performances in the field of high frequency stock market time series.

For each analysis we used the open-source software of statistics **R**.

1. Models

A *model* for prices (or returns) is a detailed description of how successive prices (or returns) are determined. A good model must describe all the potentially existing and known properties of recorded prices.

The sections following the current one will describe in a more detailed way the models we used. They can be divided into three parts: linear models, non-linear models and machine learning. Linear models include the basic exponential smoothing, the famous ARIMA time series models and an evolution of them the ARFIMA models. The non-linear models are characterized by an assumption of a non-linear function to forecast values from past values at the contrary to linear models which evidently assume a linear function to fit the values. We used an autoregressive non-linear model, generalization of AR models: the STAR models. The second non-linear model is a hybrid model taking into account the importance of volatility in finance: it's the ARMA-GARCH models. Finally more complex models are used and maybe the more accurate: machine learning. We used two different neural networks since they have found some predictability capacities in detecting recurrent non-linearity patterns. The first neural network used as inputs five past values, so this one is an evolution of the former models. The second one tried to implement the market dynamics with open, high, low, close prices and volume of the current period.

2. Data pre-processing

In order to forecast stock prices there are different ways to begin. Direct statistical analysis of financial prices is difficult, because consecutive prices are highly correlated and the variances of prices increase with time as we have seen in section II. Prices are not stationary, a concept also introduced in the same section. Consequently, it is more convenient to analyse changes in prices. Results for changes, for example a forecast, can easily be used to give appropriate results for prices. Suppose the price is recorded once on each intraday time (each 15 minutes), always at the same time of days. We used for the most part of the models the closing prices, only machine learning used other prices. Let z_t be the price on trading t and let d_t be the dividend (if any) paid during day t; d_t will only be non-zero for stocks and then only on a few days every year. Three types of price changes have been used in previous research:

$$X_t^* = z_t + d_t - z_{t-1} \qquad \textbf{(III.1)}$$

$$X_t = \log (z_t+d_t) - \log (z_{t-1}) \qquad \textbf{(III.2)}$$

$$X_t' = (z_t + d_t - z_{t-1}) / z_{t-1} \qquad \textbf{(III.3)}$$

In our case, as we use high frequency stock prices and as we want to reduce the non-pertinent complexity, d_t equals zero. The first difference x_t^* depend on the price units, so comparisons between series are difficult. They have the further disadvantage that their variances are proportional to the price level. For these reasons, either the x_t or the x_t' are nearly always studied in modern research. In our case, we used the three different returns and chose the model with which the forecasting performances were the best.

We also tried a standard/mean scaling (z-index). While linear scaling strictly preserves the uniformity of the distribution, mean scaling helps in creating a more uniform distribution, which can be desired if most of the extreme values of the original distribution are considered to be noisy and that is our case like we can see in the box plots of our times series.

3. One-step-ahead predictions

To evaluate the different forecasting performances of our models we used one-step-ahead predictions. It means that we only forecast the just-next value of one present value that is in our case the value at +15 minutes. One-step-ahead forecasting can prevent problems associated with cumulative errors from the previous period for out-of-estimation sample forecasting. This is the advantage that also justifies our choice. Furthermore, since we use high frequency data, the possible application of our study is high-frequency trading, so short term trading and predictions: from this point of view, one-step-ahead predictions seem adequate. Moreover, some models we use not seem appropriate to multiple-step predictions like ARIMA models.

4. Training and validation sets

To use our models we first have to train them, which means calculate and determine the different parameters and the different coefficients of the models, so that they could learn the rules which could determine the future value from past values. For this first part of training we have to use a training set. This is our *in-sample set*. After this procedure, a validation of the model is used, to avoid overfitting. Overfitting is a characteristic of a model that implies a too high importance for the training data, which means that the model is too specialized and it cannot generalize to new data. The new data are the validation set: the *out-sample set* which validates a model, prefers a model to another. To avoid over-accuracy the in-sample and out-sample have to be disjoined.

The out-sample set is also used to evaluate the forecasting performances of the different types of models.

Our whole dataset is composed of 1600 data that is 1600 open, high, low prices, transaction volume and above all close prices. The training set is composed of 1500 data and the validation set of 100 data which means a percentage of 6% for the validation set.

5. Recursive loop and output for machine learning

As we decided to focus on a one-step-prediction, a good idea is to use a loop which can allow us to use all data we can have: some recursive loops have been implemented in the code of our models. For the linear and non-linear models the training set was added by the first value of the validation set and so on: the training phase was thus done 100 times. The coefficients were so calculated at each step of the validation set: we used the best model at time t to improve our forecasting performances.

For the machine learning, using a loop is too time-expensive versus a gain of forecasting accuracy. The machine learning actually learns well and it is enough to use as output the price at t+1 to determine it and to use the entire validation set during the test phase, without training the model again. At this time, the determination of the coefficients of the model was done once.

6. Modelling benchmark

To evaluate the forecasting ability of our models, we use the random walk model (RW) as a benchmark for comparison. RW is a one-step-ahead forecasting method, since it uses the current actual value to predict the future value as follows:

$$f_{t+1} = y_t \qquad \textbf{(III.4)}$$

Where y_t is the actual value in the current period t and f_{t+1} is the predicted value in the next period.

7. Forecasting evaluation and accuracy measures

After choosing and determining the models with the training set, we used them to make forecasts. The 100 one-step-ahead forecasts were calculated (with the methods we saw below) and we compared them to the real values from the validation set to evaluate the forecasting performances of one model. To do this we use statistical indicators of good fit and an economic indicator.

e. Statistical Indicators of good fit

The statistical indicators use the forecasting values and the real values to calculate a function of them which determine the error term. Evidently the best model will be the one with the lowest statistical indicators to minimize the errors terms from the forecasting values and the real values. The next table shows the forecasting indicators mostly used. The value e_t is the error term (difference between forecasting value f_t and real value y_t: f_t - y_t). All of these indicators calculate a mean of the error term along the validation set. The most common is the MSE. In statistics, the mean square error or MSE of an estimator is one of many ways to quantify the difference between an estimator and the true value of the quantity being estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. MSE measures the average of the square of the "error." The error is the amount by which the estimator differs from the quantity to be estimated. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate.

$$ME = \frac{1}{T}\sum_{t=1}^{T} e_t \qquad MSE = \frac{1}{T}\sum_{t=1}^{T} e_t^2 \qquad MAE = \frac{1}{T}\sum_{t=1}^{T} \left|e_t\right|$$

$$MPE = \frac{1}{T}\sum_{t=1}^{T} 100 \times \left(\frac{e_t}{y_t}\right) \qquad MAPE = \frac{1}{T}\sum_{t=1}^{T} 100 \times \left(\frac{\left|e_t\right|}{y_t}\right)$$

$$U_1 = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - f_t)^2}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T} y_t^2} + \sqrt{\frac{1}{T}\sum_{t=1}^{T} f_t^2}}$$

$$U_2 = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T-1}\left(\frac{f_{t+1} - y_{t+1}}{y_t}\right)^2}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T-1}\left(\frac{y_{t+1} - y_t}{y_t}\right)^2}}$$

Table 4: Performance indicators

- A low value of the ME may conceal forecasting inaccuracy due to the offsetting effect of large positive and negative forecast errors. However, despite the unbiasedness of the forecasts, their inaccuracy becomes apparent from inspection of subsequent forecast evaluation statistics.
- The MSE and MAE may overcome the "cancellation of positive and negative errors" limitation of the ME, but they fail to provide information on forecasting accuracy relative to the scale of the series examined: Consideration in this case of the scaled measures (MPE, MAPE, $U_{1,}$ and U_2).
- The MSE places a greater penalty on large forecast errors than the MAE. We can also use the RSME taking the root of the MSE to diminish this

characteristic. In an analogy to standard deviation, taking the square root of MSE yields the root mean squared error or RMSE, which has the same units as the quantity being estimated. The root mean square deviation (RMSD) or root mean square error (RMSE) is a frequently-used measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modelled or estimated. RMSD is a good measure of precision. These individual differences are also called residuals, and the RMSD serves to aggregate them into a single measure of predictive power.

- The more accurate the forecasts, the lower the value of the U_1 statistic. The U_1 statistic is bounded between 0 and 1, with values closer to 0 indicating greater forecasting accuracy. This is an indicator of distance between forecast and target values.
- The U_2 statistic will take the value 1 under the naive forecasting method (see III.6 with the Random Walk modelling benchmark). Values less than 1 indicate greater forecasting accuracy than the naive forecasting method; values greater than 1 indicate the opposite.
- Even the simplest forecast evaluation statistic provides useful information. While the more sophisticated forecast evaluation statistics provide information on the properties of the alternative forecasts, the mean error provides useful information on the bias of the actual forecast errors.

U_2 is also called Theil’s U. This is the one we used here to compare the forecasting performances of our models. The Theil’s U indicator has to be minimized. Indeed this indicator has two advantages:

- Its numerator takes into account a “percentage” root mean squared error (like a RPMSE) with the advantages of the “percentage”, “squared” and “root” peculiarities we saw earlier. A minimal Theil’s U will be linked with a minimal RMSE.
- By its definition, it can be easily used to compare it with our benchmark, the naïve prediction, and to see which model can outperform at best the random walk. The traditional indicators are indicators of fitting and since our goal in the financial field is to beat the random walk and the naïve prediction, this indicator is the most appropriate in finance.

f. An economic indicator: HIT rate

A statistical indicator is not enough in the field of financial forecasting and particularly in one-step-ahead predictions. Indeed, since our possible application is high-frequency trading, our great interest is if the next price value will increase or decrease in order to buy it or sell it. So what we are interesting in is the sign of the predicted return. A good model predicts the positive or negative sign better than a naïve model so better than 50% (since with the law of large numbers, if you predict the sign of returns randomly you will arrive at a percentage of 50)

This indicator is called HIT which is the percentage of good price movements (good signs of return) predictions, it is defined as follows:

HIT = (number of signs predicted) / (length of validation set) * 100 **(III.5)**

The HIT indicator is an important indicator for a one-step-ahead prediction since we cannot validate a model with a good Theil’s U and with a bad HIT; because no profit could be made with a HIT<50. Actually the HIT and the Theil’s U are complementary indicators since we cannot make profits with a high HIT and a high Theil’s U since the possible loss could be too great and the potential gain could be biased. Without implementing a trading strategy, the validation of the model is a trade-off between the statistical Theil’s U indicator of fitting and the economic profitable indicator HIT rate.

IV. Linear Models

Time series, of whatever type they are – financial, economic or from historical sales – have to be first modelled by the "simplest", but not the worst, models existing for the fitting and for the prediction: the linear models. Linear models try to fit at best the present value of a given time series, given its past values, with a linear function. In this part of the thesis we will discuss three models. As we said before, the financial time series are known to be non-stationary, so there can be a trend, up or down, depending on if we are in a bull market or in a bear market. That's why the models used here have to take into account this characteristic. We thus assume in this part that financial time series can be explained and forecasted linearly by their past values.

The three models studied here will be: the exponential smoothing with or without trend correction, which is the Holt-Winters model without the seasonality component because, as we have seen, there is no seasonal component; then the autoregressive integrated moving average (ARIMA) and the autoregressive fractionally moving average (ARFIMA). The "integrated" part for the two last models is essential to capture and eliminate the non-stationary characteristic.

1. Holt-Winters model

Despite their simplicity, exponential smoothing models constitute very polyvalent and accurate predictive methods for time series analysis, and they were showed as very suitable methods to economic phenomenon, we can motion for this the work of De Gooijer et al. in 2006 *25 years of time series forecasting* [1]. Firstly formulated on empirical and intuitive bases, they have found theorical justifications. They were named as Holt-Winters for the two scientists Holt and Winters who have developed these models.

Exponential smoothing is a technique that can be applied to time series data, either to produce smoothed data for presentation, or to make forecasts. The time series data themselves are a sequence of observations. The observed phenomenon may be an essentially random process, or it may be an orderly, but noisy, process. Whereas in the simple moving average the past observations are weighted equally, exponential smoothing assigns exponentially decreasing weights over time. And it is particularly suitable and interesting for our case since we can easily imagine that most recent values have more effects and are more useful for the prediction of the next stock value.

a. Basic exponential smoothing

Exponential smoothing is commonly applied to financial market and economic data, but it can be used with any discrete set of repeated measurements. The raw data sequence is often represented by $\{x_t\}$, and the output of the exponential smoothing algorithm is commonly written as $\{s_t\}$ which may be regarded as our best estimate of what the next value of x will be. When the sequence of observations begins at time $t = 0$, the simplest form of exponential smoothing is given by the formulas:

$$s_1 = x_0$$
$$s_t = \alpha x_{t-1} + (1-\alpha)s_{t-1}, t > 1 \qquad \textbf{(IV.1)}$$

Where α is the smoothing factor, and $0 < \alpha < 1$. In other words, the smoothed statistic s_t is a simple weighted average of the previous observation x_{t-1} and the previous smoothed statistic s_{t-1}. If α coefficient is close to one then the model gives a major weight to most recent observations otherwise the model is a little “inert”, indeed it actually gives a uniform weight to all past values. The prediction for the next value of x, x_t, is s_t.

The simple exponential smoothing model is not suitable for non-stationary process with trend component because it is still late compared to the real time series values and produce biased predictions. The book *Business Intelligence* [28] of Carlo Vercellis insists on this characteristic.

b. *Exponential smoothing with trend adjustment*

To take into account the non-stationary characteristic with trend, so to suit at best a smoothing model to a financial time series with trend, we have to use the extended model which can correct it. In such situations, double exponential smoothing can be used.

Again, the raw data sequence of observations is represented by $\{x_t\}$, beginning at time $t = 0$. We use $\{s_t\}$ to represent the smoothed value for time t, and $\{b_t\}$, newly introduced sequence, is our best estimate of the trend at time t. The output of the algorithm is now written as F_{t+m}, an estimate of the value of x at time t+m, m>0 based on the raw data up to time t. In our case, m is equal to one according to our forecasting methodologies described in the third chapter. Double exponential smoothing is given by the formulas:

$$\begin{aligned} s_0 &= x_0 \\ s_t &= \alpha x_t + (1-\alpha)F_t \\ b_t &= \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1} \\ F_{t+m} &= s_t + m b_t, \end{aligned} \qquad \textbf{(IV.2)}$$

Where α is the data smoothing factor, 0 < α < 1, like before and β, a new parameter, is the trend smoothing factor, 0 < β < 1.

If α coefficient is close to one then the model gives a major weight to most recent observations. When β coefficient is close to zero the model gives an almost uniform weight to trends. On the other hand when it is close to one the most recent trend is dominant.

Unknown parameters α and β (for the second model) are determined by minimizing the squared prediction error.

c. *Forecasting method and R implementation*

The way to choose the best Holt-Winters models is described here:

Step 1:

Data pre-processing; for this model the time series used were the stock prices themselves without using the returns, it was actually more accurate.

Step 2:

The best Holt-Winters model fitting our financial time series was chosen by minimizing the squared errors using the training set. Then the two parameters α and β of the model were determined.

Step 3:

It was then applied to predict the one-step-ahead value. The training set was added by the next value (taken in the validation test) and the best model was again fitted like in step 2 to go then to step 3. This procedure was used until the validation set was completely used.

Step 4:

For one time series, we chose the best model between the simple exponential smoothing and the double exponential smoothing, using the indicators of fitting goodness for the validation set.

This calculation was done for the entire validation set to give, as output, 100 one-step-ahead predictions. The R function *HoltWinters()* was used to determine the two parameters of the model.

Regarding our study, the simple exponential smoothing outperforms the double exponential smoothing for almost cases.

2. ARIMA model

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model, well used for non-stationary time series thanks to, as we said, the time series values differentiation.

We will try here to fit and forecast our financial time series with an ARIMA model. We will describe the general model, and then the way used to choose the best model and to finish with the implementation in R. Whereas exponential smoothing is, we can say, a data extrapolation, the ARIMA model try to find a sort of best regression between past values and the actual values. Past values could be thus explicative variables to the future value using a linear function. We can expect that the ARIMA models are more accurate than exponential smoothing since they try to find the best adaptive linear function whereas the functional form is imposed in Holt-Winters model; there is more flexibility on the parameters for the ARIMA models. This model is at the centre of Box and Jenkins time series modelling methodology.

a. Model description

As an ARMA(p, q), an ARIMA(p, d, q) process has an autoregressive part with order p (on the left side of the following formula) and a moving average part with order q (on the right side), the parameter d is for the integrated part. The parameters p, d and q are integer values greater than or equal to zero for this type of model. The values of the parameters depend on the model we want to use.

Given a time series of data X_t where t is an integer index and the X_t are real numbers, an ARIMA(p, d, q) is given by:

$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)(1-L)^d X_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t \qquad \textbf{(IV.3)}$$

Where L is the lag operator for time series differentiation, the ϕ_i are the coefficients of the autoregressive part of the model, thus the regression part with time series past values, the θ_i are the coefficients of the moving average part thus the regression part between the actual value and passed error terms ε_{t-i}: the error terms are valued from the difference between the fitting values and the real values. ε_t is a white noise, and is generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean. In our case, the time series X_t will be of course the stock market closing price.

ARIMA models are used for observable non-stationary processes X_t that have some clearly identifiable trends:

- constant trend (i.e. a non-zero average) leads to d = 1
- linear trend (i.e. a linear growth behaviour) leads to d = 2
- quadratic trend (i.e. a quadratic growth behaviour) leads to d = 3

In our cases, some financial time series can have a significant trend so d=2 is sometimes appropriate. Since the financial times series studied here are for all of them non stationary, a d parameter equal to one is necessary.

The ARIMA models with stock prices outperformed the ARMA models using returns. So it is this way we looked into.

b. *Steps to the best model*

The method to choose the best ARIMA model for one time series was here a mix between different approaches. We also tried the Box and Jenkins methodology based on the autocorrelogram and the partial autocorrelogram to determine the model parameters but our method outperformed this method. Indeed sometimes the Box and Jenkins gave equivalent results to the ones with the method we used but sometimes not. There are also some functions in R which can automatically find p and q, like *auto.arima()* but these methods were less accurate than the one we use.

Step 1:

The first step was to choose with the training sample, different models with essential characteristics which we will see below. The θ_i and ϕ_i coefficients of the fitting model were determined with the Maximum Likelihood. Different models, with different parameters p, d and q, were tested. The d parameter was selected to make the time series stationary. Usually d was equal to one. Then we tried several models with different p and q parameters, we tried even so only autoregressive models (q=0) or only moving-average models (p=0). Nevertheless the models tested had to be the simplest to avoid overfitting, so we restricted a maximum value of 4 for each p and q. We chose the models with a p-value for the fitted coefficients inferior to 0.05 to accept the null hypothesis test that the coefficients were significantly different from zero.

Step 2:

Then we compare the AIC criterion, the Akaike information criterion:

$$AIC = 2k - 2\ln(L) \tag{IV.4}$$

Where k is the number of parameters in the statistical model which is the sum of p and q for an ARIMA and L is the maximized value of the likelihood function for the estimated model. This criterion has to be minimized, so the models with an AIC sufficiently low were selected.

Step 3:

During these steps, the errors terms were analysed. We analysed their autocorrelation diagrams (ACF), their standardized plots and their p-values of Ljung-Box test; all of this is to check if the error terms were a white noise with zero mean: essential characteristic to select a model. Here follows the typical white noise graphics:

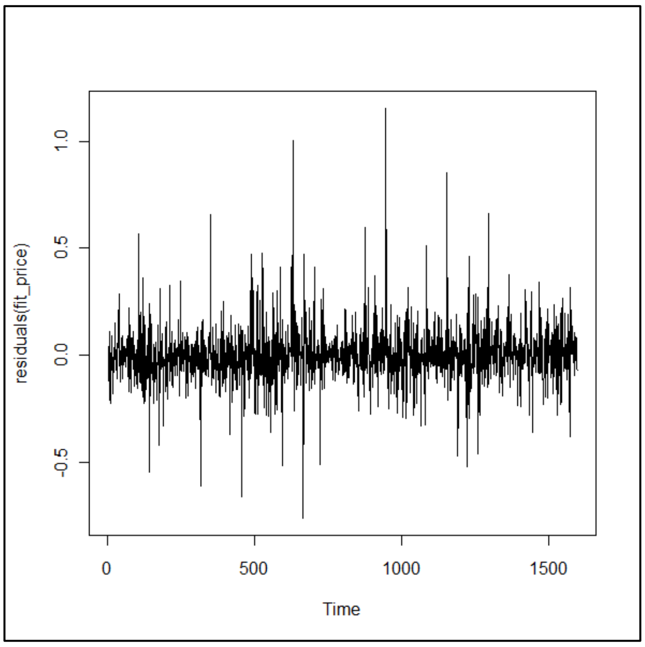


Figure 5: Standardized error terms as white noise

We can see on this first plot the first characteristic of standardized error terms as a white noise: mean equal to 0 and constant variance.

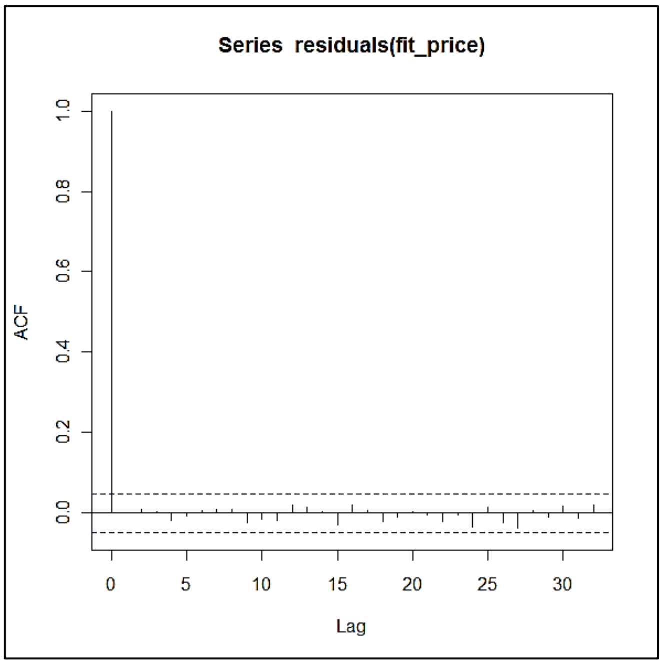


Figure 6: Autocorrelation diagram (ACF) for a white noise

The different values of autocorrelations have to be inferior to 0.05, maximum value in order to consider errors terms not correlated. The figure 6 shows it.

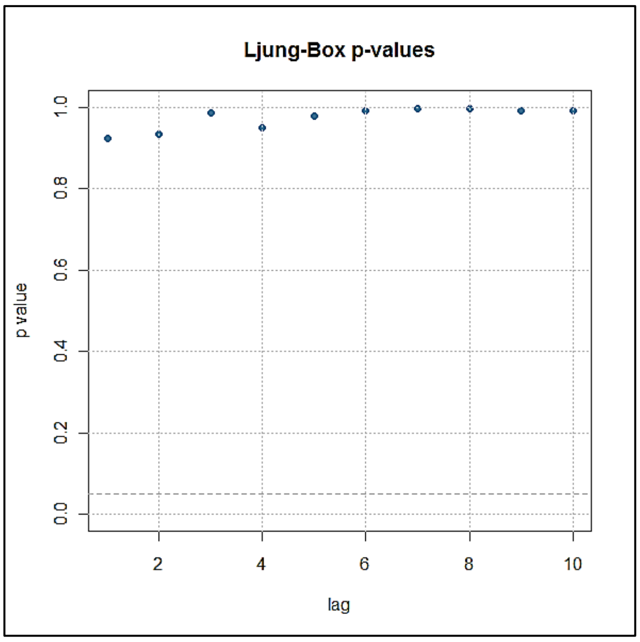


Figure 7: Ljung-Box test for white noise

The Ljung–Box test is a type of statistical test of whether any of a group of autocorrelations of a time series is different from zero. Instead of testing randomness at each distinct lag, it tests the "overall" randomness based on a number of lags. The Ljung–Box test can be defined as follows.

H_0: The data is random.

H_a: The data is not random.

For a p-value sufficiently high (>0.05), we accept the null hypothesis: errors terms are random so independent. The test statistic is:

$$Q = n\,(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k} \tag{IV.5}$$

Where n is the sample size, $\hat{\rho}_k$ is the sample autocorrelation at lag k, and h is the number of lags being tested. For significance level α, the critical region for rejection of the hypothesis of randomness is:

$$Q > \chi^2_{1-\alpha,h} \tag{IV.6}$$

Where: $\chi^2_{1-\alpha,h}$ is the α-quantile of the chi-square distribution with h degrees of freedom. In our case, h is equal to one.

The error terms must also have a normal distribution. To verify this condition, a qqplot was done to compare the distribution of the errors to the normal distribution.

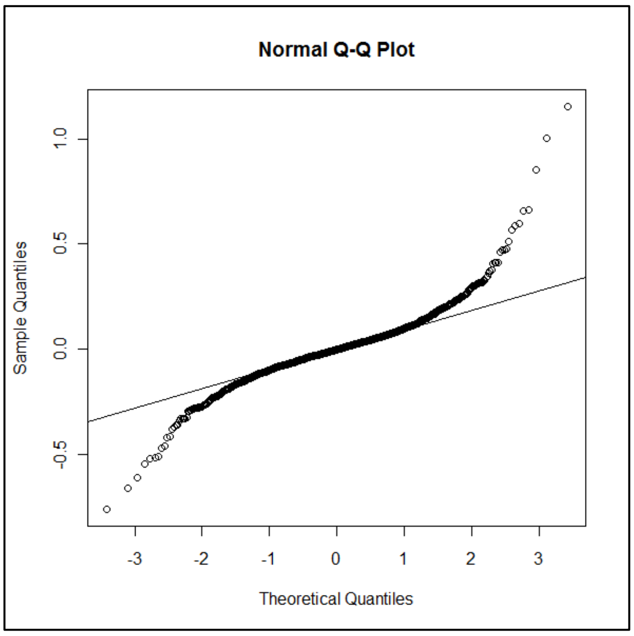


Figure 8: Normal qqplot for error terms

The errors terms can be considered as normal but we have very big tails. This is due to the highest and lowest returns we have which can be considered as outliers. This is typical from financial time series as we can see in the work of Dacorogna et al. in 2003 *An Introduction to High Frequency Finance* [5].

All of these verifications are essential for the validity of the model.

Step 4:

After selecting the models satisfying these conditions, one-step-ahead predictions were done for the whole validation set. The best model chosen and selected was then the one with the best accuracy indicators: this is our ARIMA model with its parameters p, d and q. In each case, p and q were superior or equal to one.

c. Implementation in R

The R code for the ARMA modelling used the command *armaFit()* from the package *fArma* to fit the coefficients for a given model after selecting the parameters with our different steps. After specifying the model with the training set, the predictions were done one-step-ahead with the validation set thanks to the package *forecast*. The training set was added at every step by the next value from the validation set.

3. ARFIMA Model

The ARFIMA model is an extension and generalization of the ARIMA model since it allows the d parameter for the differentiation to be non-integer and irrational or even non-positive. The acronym ARFIMA stands for Autoregressive Fractionally Integrated Moving Average. They are useful in modelling time series with long memory. Long memory may cause a slower decay of the autocorrelation function than would be implied by ARMA models. In other words, long-range dependency (abbreviated as LRD) is a phenomenon that may arise in the analysis of spatial or time series data. It relates to the rate of decay of statistical dependence, with the implication that this decays more slowly than an exponential decay like we can see if the figure 9 above:

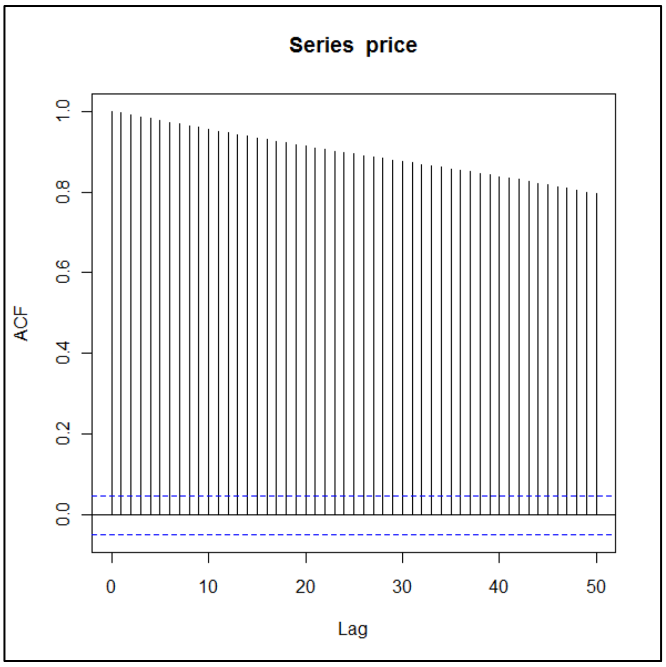


Figure 9: Long rate of decay for ACF

a. Model description

Fractional integration first appears in literature in the studies of *Granger and Joyeux (1980) and Hosking (1981).* The model, known as Autoregressive Fractionally Integrated Moving Average (ARFIMA), allows for increased flexibility in modelling high-frequency dynamics. ARFIMA model is written as follows:

$$\left(1-\sum_{i=1}^{p}\phi_i B^i\right)(1-B)^d X_t = \left(1+\sum_{i=1}^{q}\theta_i B^i\right)\varepsilon_t \, . \qquad \textbf{(IV.7)}$$

Where the d parameter is fractional integration parameter as a real number, B is lag operator and ε_t is white noise residual, as before. Polynomial structures of this equation lie outside the unit circle, satisfying the stationarity and invariability conditions. The fractional differencing lag operator $(1- B)^d$ is defined by the binomial expansion as follows:

$$\begin{aligned}(1-B)^d &= \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k \\ &= \sum_{k=0}^{\infty} \frac{\prod_{a=0}^{k-1}(d-a)\ (-B)^k}{k!} \\ &= 1 - dB + \frac{d(d-1)}{2}B^2 - \cdots \, .\end{aligned} \qquad \textbf{(IV.8)}$$

ARFIMA process is nonstationary when $|d| \geq 0.5$. Stationarity and invertibility conditions require that the value of the fractional differencing parameter, d, is such that $|d| < 0.5$. For the values $0 < d < 0.5$, the process ARFIMA is stationary and long term dependent, which means that is said to exhibit long memory. The process exhibits short memory for $d = 0$ (and is as an ARMA model) and intermediate memory for $d < 0$.

b. *Determination of the d parameter*

The difference between an ARIMA model and an ARFIMA can be viewed as conceptual. Indeed the d coefficient adds one parameter in the model and has to be determined. Since d can be a real value we cannot find it alone like we did for the parameters of the ARIMA, we must use an optimization algorithm.

The d parameter can be determined by different algorithms. We can mention the method of Geweke and Porter-Hudak (GPH) or the Whittle estimator (WHI), which both use the spectral density. Concerning our study we use the *fracdiff* function from the package *fracdiff* in R which finds the different parameters of the model which are p, d and q by maximizing the log Likelihood function and also its coefficients ϕ_i and θ_i.

c. *Implementation in R*

Step 1: Data pre-processing: data set used here were either the returns either the stock prices themselves. We chose the model which was the most accurate. Sometimes returns also showed long decay for the autocorrelations.

Step 2: Like we described in the forecasting methodologies, we implement here an algorithm to find at every step on the validation set the best ARFIMA model. We use for this the *arfima()* function from the *forecast* package in R to fit the time series, so to determine the parameters (p, d and q) and the coefficients (ϕ_i and θ_i) and then do a one-step prediction.

Step 3: The training set was then added by the first value of the validation set and so on to have 100 one-step-ahead predictions.

V. Non Linear Models

Financial time series prediction deals with the task of modelling the underlying data generation process using past observations and using the model to extrapolate the time series into the future. Due to its intrinsic difficulty and widespread applications, much effort was devoted in the past few decades to the development and refining of financial forecasting models. In the literature, two major classes of models were studied by econometricians for the purpose of forecasting. They are the statistical time series models and structural econometric models. Linear time series models such as the Box-Jenkins autoregressive integrated moving average (ARIMA) models were among the first to be developed and subsequently widely studied and we studied them in the previous chapter. Despite its simplicity and versatility in modelling several types of linear relationship such as pure autoregressive, pure moving average and autoregressive moving average (ARMA) series, such type of models was constrained by its linear scope. As we have seen in section II, non-linearities in financial time series can be present and as a result models have to adapt themselves to this characteristic. That is why it is a good opportunity to try non-linear models which can detect and learn these particularities in time series. Machine learnings are good models to detect non-linearities as a result of their intrinsic methodology.

There can be different approaches about implementing non-linear models. We can try a non-linear model which could generalize the linear models: it will the STAR model or a machine learning which has as inputs past values. Non-linear particularities can be surely explained by market dynamics which characterize its running: the offer and demand dynamics. We can implement it by using machine learning with input the open, high, low, close prices of an instant t and the transaction volume.

A well-known non-linear market characteristic concerns its volatility: this can be well-described by a GARCH model which can predict the volatility of a stock or of the market. To forecast a stock value we have to use a hybrid model using a GARCH for the volatility part and an ARMA for the price value part. It is the ARMA-GARCH and we will begin this section by a description of this model to follow with the STAR model. Machine learning will be seen on the next section.

Many papers have also shown that financial markets were characterized by a non-linear behaviour like we have seen it in the state of art in section I, and that is auspicious for this current section.

1. An hybrid ARMA model with volatility: the ARMA-GARCH

a. Motivation: volatility clustering

In finance, volatility most frequently refers to the standard deviation of the continuously compounded returns of a financial instrument within a specific time horizon. It is used to quantify the risk of the financial instrument over the specified time period. When volatility is high, the risk is high too.

It's common knowledge that types of assets experience periods of high and low volatility. That is, during some periods prices go up and down quickly, while during other times they might not seem to move at all.

Periods when prices fall quickly (a crash) are often followed by prices going down even more, or going up by an unusual amount. Also, a time when prices rise quickly (a bubble) may often be followed by prices going up even more, or going down by an unusual amount.

The converse behaviour, “doldrums” can last for a long time as well. Most typically, extreme movements do not appear “out of nowhere”; they are presaged by larger movements than usual. This is termed autoregressive conditional heteroskedasticity. Of course, whether such large movements have the same direction, or the opposite, is more difficult to say. And an increase in volatility does not always presage a further increase - the volatility may simply go back down again. This characteristic is well modelled by ARCH and GARCH models for the volatility in finance. In finance, volatility clustering refers to the observation, as noted by Mandelbrot (1963), that "large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes". A quantitative manifestation of this fact is that, while returns themselves are uncorrelated, absolute returns $|r_t|$ or their squares display a positive, significant and slowly decaying autocorrelation function: $cor(|r_t|, |r_{t+\tau}|) > 0$ for τ ranging from a few minutes to a several weeks.

Observations of this type in financial time series have led to the use of GARCH models in financial forecasting and derivatives pricing. The ARCH (Engle, 1982) and GARCH (Bollerslev, 1986) models aim to more accurately describe the phenomenon of volatility clustering and related effects such as kurtosis. The main idea behind these two widely-used models is that volatility is dependent upon past realizations of the asset process and related volatility process. This is a more precise formulation of the intuition that asset volatility tends to revert to some mean rather than remaining constant or moving in monotonic fashion over time.

Autoregressive Conditional Heteroskedasticity (ARCH) models are used to characterize and model observed time series. They are used whenever there is reason to believe that, at any point in a series, the terms will have a characteristic size, or variance. In particular ARCH models assume the variance of the current error term or innovation to be a function of the actual sizes of the previous time periods' error terms: often the variance is related to the squares of the previous innovations. ARCH models are employed commonly in modelling financial time series that exhibit time-varying volatility clustering, i.e. periods of swings followed by periods of relative calm like we said before. In statistics, a sequence of random variables is heteroscedastic, or heteroskedastic, if the random variables have different variances. The term means "differing variance" and comes from the Greek "hetero" ("different") and "skedasis" ("dispersion"). In contrast, a sequence of random variables is called homoscedastic if it has constant variance. Here follows the typical ARCH (q) model:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \qquad \textbf{(V.1)}$$

Where ε_t can be the returns themselves (for an ARCH to predict the volatility of a stock return) or the errors terms taken from an ARMA model for the returns. The α_i coefficients are the coefficients of the model.

The GARCH model, like its name said, generalize the ARCH model allowing the process to be linked with the previous variances like an autoregressive model. It is similar between ARMA and MA models. Here follows the description of a GARCH (p, q) model **(V.2):**

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$$

Where ε_t can be the returns themselves (for a GARCH to predict the volatility of stock return) or the errors terms taken from an ARMA model for the returns. The α_i coefficients are the coefficients of the model for the ARCH part. As we can see on this formula we added an autoregressive part with the past variances; the β_i are the coefficients of this part. The non-linearity characteristic is evidently apparent with the squares to form the variances and used on the errors terms

b. Description

The mix ARMA-GARCH can be used to improve the simple ARMA model when the error terms show heteroskedasticity. Indeed the White test or Teräsvirta test can help that one should or not neglect non-linearity in a given time series, like we saw on section two.

The ARMA (p, q) – GARCH (p, q) can be described as follows. Specifically, each component of the mixture model can be denoted as a normal ARMA(p,q) series:

$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)(1-L)^d X_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t \qquad \textbf{(V.3)}$$

Furthermore, each residual term ε_t is assumed Gaussian white noise with variance denoted by the GARCH (p, q) model with this formula **(V.4)**:

$$\sigma_t^2 = \alpha_0 + \alpha_1\epsilon_{t-1}^2 + \cdots + \alpha_q\epsilon_{t-q}^2 + \beta_1\sigma_{t-1}^2 + \cdots + \beta_p\sigma_{t-p}^2 = \alpha_0 + \sum_{i=1}^{q}\alpha_i\epsilon_{t-i}^2 + \sum_{i=1}^{p}\beta_i\sigma_{t-i}^2$$

For a theorical view about the optimization algorithm to determine the coefficients of the models and so the one-step-ahead prediction value, the lector could refer to [16]: see derivation of the Generalized Expectation-Maximization (GEM) algorithm for implementation.

c. *Implementation in R*

For this model we used the recursive loop like we did for the previous models. The *garchFit()* from the *fGarch* package was able to implement the ARMA-GARCH, calculating the coefficients and forecasting one-step-ahead.

Step 1:

Testing the residuals of the ARMA models found on section II with the white test to test the heteroskedasticity and to see previously if the ARMA-GARCH could be appropriate. The ARMA-GARCH was by the way implemented for each stock.

Step 2:

Determination of the parameters of the model that is the different p and q of the ARMA part and GARCH part. For this step we tried different models training them. We chose the one which had the best forecasting performances for the validation set.

Step 3:

We verified that the coefficients of the models were for the major part significantly different from zero, that the AIC criterion was sufficiently low and the residuals satisfied a white noise characteristic as for the ARMA model in section IV.

d. Heteroskedasticity of ARIMA error terms

We tested the error terms of the ARIMA to show or not the presence of heteroskedasticity with the white test. Here are our results with the p-value:

Table 5: p-values of heteroskedasticity of errors terms

	[1]	**[2]**	**[3]**	**[4]**	**[5]**
p-value	0.3295	0.6191	0.2876	0.0334	0.0492

	[6]	**[7]**	**[8]**	**[9]**	**[10]**
p-value	0.0117	0.3038	0.8210	0.0011	0.3956

We can see that not each error term shows heteroskedasticity (the null hypothesis of the white test is linearity in mean). But when heteroskedasticity is found in the ARIMA error terms, you will see that the ARMA-GARCH model can improve the forecasting performances of the ARIMA.

2. STAR model as extension of ARIMA: a non-linear autoregressive model

a. Motivation

To exploit non-linearities possibly present in our high frequency financial time series we can use a novel model, an evolution of the linear autoregressive model. These non-linearities were presented on section two. We saw thanks to appropriate tests that some time series could be defined as non-linear, sometimes for stock prices themselves, sometimes for returns. So this model would have either data pre-processing or no one. To capture these non-linearities one way to begin is to use a non-linear model for forecasting.

Smooth Transition Autoregressive (STAR) models are typically applied to time series data as an extension of autoregressive models, in order to allow for higher degree of flexibility in model parameters through a smooth transition.

Given a time series of data x_t, the STAR model is a tool for understanding and, perhaps, predicting future values in this series, assuming that the behaviour of the series changes depending on the value of the transition variable. The transition might depend on the past values of the x series, or exogenous variables. In our case, only past values are in the study.

The model consists of 2 autoregressive (AR) parts linked by the transition function. The model is usually referred to as the STAR(p) models proceeded by the letter describing the transition function (see below) and p is the order of the autoregressive part. Most popular transition functions include exponential function and first and second-order logistic functions. They give rise to Logistic STAR (LSTAR) and Exponential STAR (ESTAR) models. The advantage of these models in R is the previous test of non-linearity they assume. This type of model is also called regime switching model because of the switch transition function. The two AR parts implies that there are two regimes. These models imply the existence of regimes with potentially different dynamic properties, but with a smooth transition between regimes (e.g., Granger and Teräsvirta, 1993; Teräsvirta et al., 1994).

b. Model description

Consider a simple AR(p) model for a time series y_t where γ_i for i=1, 2, ..., p are autoregressive coefficients, assumed to be constant over time. $\epsilon_t \sim^{iid} WN(0;\sigma^2)$ stands for white-noise error term with constant variance. Written in a following vector form:

$$y_t = \mathbf{X_t}\gamma + \sigma\epsilon_t. \qquad \textbf{(V.5)}$$

Where: $\mathbf{X_t} = (1, y_{t-1}, y_{t-2}, \ldots, y_{t-p})$ is a column vector of variables γ is the vector of coefficients: $\gamma_0, \gamma_1, \gamma_2, \cdots, \gamma_p$ and $\epsilon_t \sim^{iid} WN(0;1)$ stands for white-noise error term with unit constant variance.

STAR models were introduced and developed by Kung-sik Chan and Howell Tong in 1986, in which the same acronym was used. It originally stands for Smooth Threshold Autoregressive. The models can be thought of in terms of extension of autoregressive models, allowing for changes in the model parameters according to the value of weakly exogenous transition variable z_t.

Defined in this way, STAR model can be presented as follows:

$$y_t = \mathbf{X_t}\gamma^{(1)}G(z_t,\zeta,c) + \mathbf{X_t}\gamma^{(2)}(1 - G(z_t,\zeta,c)) + \sigma\epsilon_t \qquad \textbf{(V.6)}$$

As we can see it is a sort of non-linear autoregressive model where:

y_t are the returns or stock prices, $\gamma^{(1)}$ and $\gamma^{(2)}$ (for i=0,1,2…p and j=0,1,2,. . ., p) are the unknown coefficients that correspond to each of the two regimes, so each of the AR parts and $\epsilon_t \sim^{iid} WN(0;1)$.

$G(z_t, \zeta, c)$ is the transition function, assumed to be twice differentiable and bounded between 0 and 1, ζ is the transition rate or smoothness parameter, c is the threshold value which represents the change from one regime to another and d is the number of lags of the transition variable. This function introduces regime switching and nonlinearity into the parameters of the model. The transition variable, z_t, is usually (but not always) defined as a linear combination of the lagged values of y_t which means that the transition variable uses the information taken from the past values until a lag of d. Regarding the choice of transition function, the two most widely used in the literature are the first-order logistic function $G(z_t, \zeta, c) = (1 + exp(- \zeta(z_t - c))^{-1}$ $\zeta > 0$ in which case the model is called logistic STAR or LSTAR(p ; d) and the first-order exponential function, for which $G(z_t, \zeta, c) = 1 - exp(- \zeta(z_t - c)^2)$ $\zeta > 0$ and in this case, the model is called exponential STAR or ESTAR(p ; d). Such models are estimated by quasi-maximum likelihood estimations (QMLE) to determine linear parameters $\gamma^{(1)}$, $\gamma^{(2)}$, and non-linear parameters ζ and c.

In our cases, we used the exponential transition function with two regimes for the model. Parameter p of the AR parts was equal to the one of the equivalent ARIMA found in our study and the d parameter for the transition variable was equal to one.

c. Implementation in R

For the computation on our statistical software, we used the function *star()* from the package *tsDyn*. This function calculates the different parameters for the model given the training set. We did the same steps as before: one-step-ahead predictions with value from validation set added to the training set until obtaining 100 predictions.

VI. Machine learning: Neural Networks

Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data. Whilst linear models have been the basis of traditional statistical forecasting models, their drawbacks have led to increase activity in non-linear modelling

To capture the possible non-linear recurrent patterns existing in our financial time series, good machine learning has to be used: the neural networks. Indeed many papers presented in section I. have shown that neural networks are well fitted for financial time series and stock market predictions. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data. Neural networks are non-linear models that can be trained to map past and future values of a time series and thereby extract hidden structure and relationships governing the data. That could be thus a considerable help in the financial field in order to find patterns and relationships between the next value of a stock and its past values. Neural networks, also called artificial or simulated neural networks, are composed by computing units (artificial neurons) interconnected so that each neuron can send and receive signals to or from others. Neural networks are a good answer for the modelling of distributed nonlinear systems as financial market. As their design is based on the human brain, they were made to acquire knowledge through learning. The process of learning for a neural network consists in adjusting the weights of each of its nodes considering the input of the neural network and its expected output. This process requires the availability of a set of input data, stock quotes or related variables like volume in our case. In our specific case we used supervised learning and the Multilayer Perceptron (MLP) as a neural network. The layout of this model is just described in the next figure:

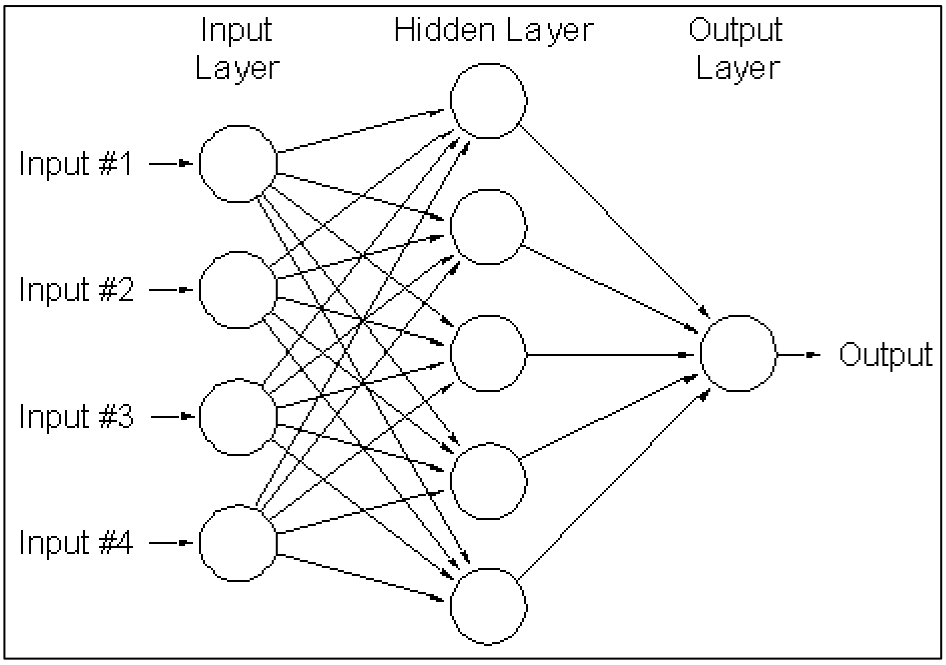


Figure 10: Layout of an MLP

The interconnected lines indicate that the value output by a cell is passed along that line to the next neuron's input stream. These are the weight values.

1. General overview

a. Supervised learning

In supervised learning, we are given a set of example pairs and the aim is to find a function in the allowed class of functions that matches the examples. In other words, we wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error (MSE) which tries to minimize the average squared error between the network's output, *f(x)*, and the target value *y* over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called Multi-Layer Perceptron, one obtains the common and well-known backpropagation algorithm for training neural networks. The MSE is the “cost function” we used here. For further details about the backpropagation algorithm the reader can see [30] (page 15-18 section 2.2.).

Mathematically, with N as the length of th training sample the cost function can be defined like:

$$\hat{C}=\frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^2 \quad \textbf{(VI.1)}$$

b. Multilayer perceptron

A multilayer perceptron (MLP) is a feed-forward artificial neural network model that maps sets of input data onto a set of appropriate output. Feed-forward means that connections between the units do not form a directed cycle but only a “one way”. This is different from recurrent neural networks. An MLP consists of multiple layers of nodes in a directed graph which is fully connected from one layer to the next.

The multilayer perceptron is a supervised neural network, by which is meant that the data used for training and testing the network is available paired with the desired response of the network, known as the target, or possibly targets for more than one output neuron. In our case we want to predict the one-step-ahead value so the output will be a single neuron and it will be the successive next price (formally output neuron = p_{t+1}). However, even for a multi-step ahead prediction, the output can take the value “p_{t+s}”

Knowledge of the desired response provides a starting point for iteratively modifying the network, by comparing the observed response with the targets and using the error MSE to drive the network’s free parameters (the weights of the different nodes, between each input neuron and each hidden neuron and between each hidden neuron and each output neuron showing by arrows on the figure 10) in a direction that will minimise the error for repeated presentations of the training input data. This is the essence of the backpropagation method, which back-propagates the errors through the network, adjusting the weights, which are free, modifiable parameters. This minimization uses an optimization algorithm to find a global minimum for the cost function. A problem that could arrive is when the optimization algorithm falls into local minima. Several training phases can be used to avoid local minima.

The training input data is assembled in the form of vectors or patterns, a collection of discrete values (also elements or variables). For each element in an input vector a corresponding input node is provided in the network input layer. For time series the input vectors will be produced by rolling a window, of some fixed length, along the series. If the task is to forecast one element ahead, like we do, then for a window of length n elements, the target will always be the (n+1)th element like we said before.

Since the network is being trained by examples, the more example input vectors available, covering the whole range of possible input data behaviour, the more accurate will be the resulting network performance. In our study for a cost-time justification, we limited the number of data in inputs that is the length of our time series. To compare with a fair value the different models we chose the same length of training set either for neural networks or previous models. The available data is divided, by an appropriate scheme discussed in detail in section III, into a training and a validation set of vectors. Like we said before, the training set is the *in-sample* set and the test set is the *out-of-sample* set.

The multi-layer perceptron neural network, in a very compact design example, is shown in the layout figure 10. The network comprises three layers of cells, with interconnections between all combinations of cell layers (adjacent cells in the same layer are not linked and connections existing only with adjacent layers).

c. *Processing*

The processing performed in the artificial neuron may be divided into four steps.

- The first step is maybe the most important since it could yield to different accuracy performances: the initial weights have to be determined randomly. And one can have to repeat the training in case of poor results.
- Data passing along input lines to a neuron are multiplied by the line weights: the process of attenuation.
- All the attenuated data inputs fanning into the hidden or output neuron are summed.
- The summation value is put through a transfer function (activation function), whose output represents the neuron’s output value.

Initially assigned random values from a pre-set range, centred on zero, the weights are incrementally adjusted during the training phase so as to achieve the desired output result for given input data. Typical weight initialisations are in the range [-1;1] or less.

For the output neuron, the transfer function was the linear function and for the hidden neurons we chose the common hyperbolic tangent sigmoid transfer function. The non-linear characteristic of the neural networks come from the hidden neurons and their non-linear activation function since from an input value in the neuron it gives a non-linearly associated value in the output. We can the plot of the sigmoid function just here:

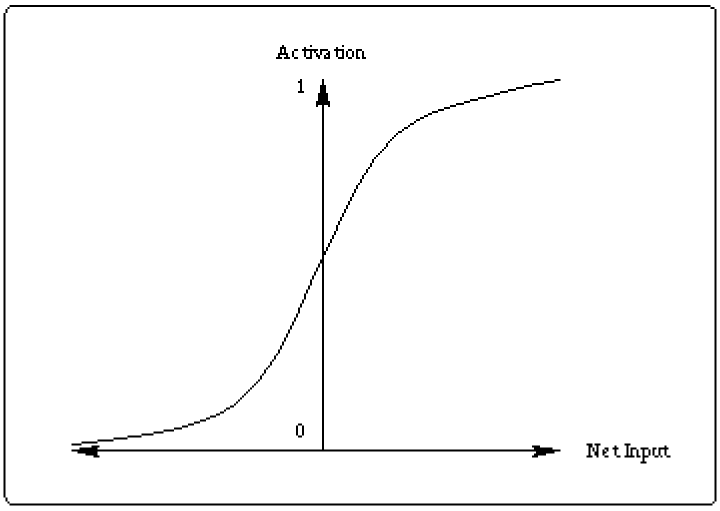


Figure 11: Sigmoid function

By plotting the qq-plot of the return and the volume we can see that they seem approximately linked by a sigmoid function which confirms our good choice :

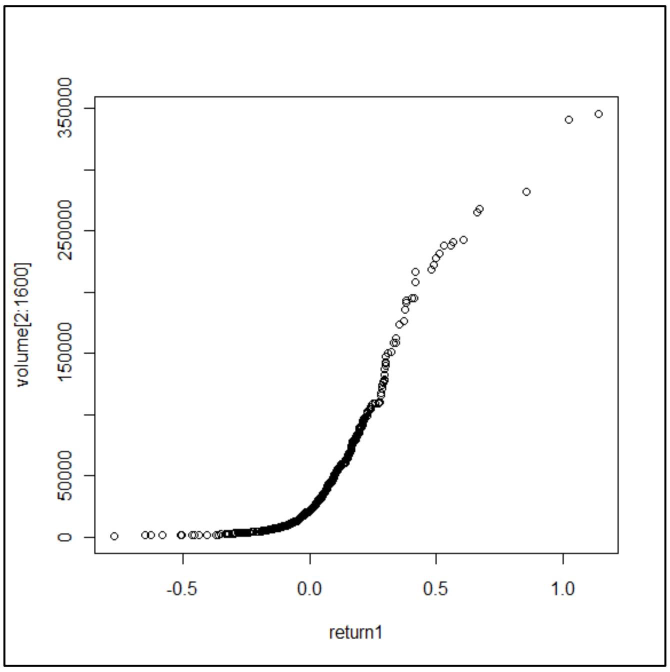


Figure 12: QQ-plot of return and volume

Mathematically, the network function f(x) (as output of a neuron) is defined as a composition of other functions $g_i(x)$ (as inputs values arriving at a neuron), which can further be defined as a composition of other functions (with a hidden layer). This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, given a neuron, where

$$f(x)=K\left(\sum_i w_i g_i(x)\right) \quad \textbf{(VI.2)}$$

Where K (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent in our case and the w_i are of course the weights of each line coming to the neuron. The sum takes into account the whole input in front of the given neuron like we explicitly see on the following figure:

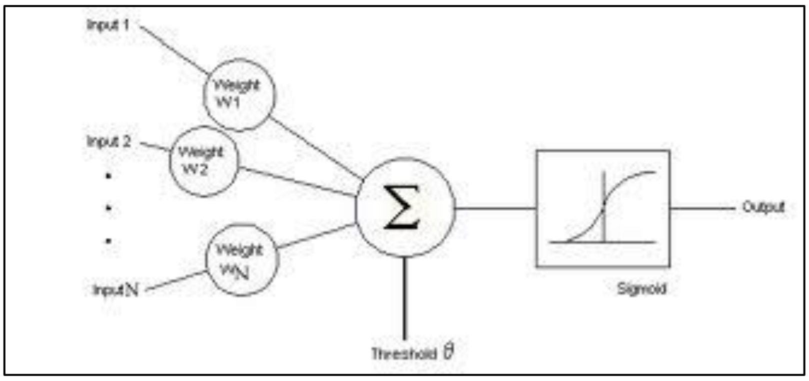


Figure 13: Processing in neural network

d. Overfitting

When one decides to use neural network, one must be careful about overfitting. Overfitting is a modelling problem when the model is too specialised and does not manage to generalise when new data come in input. With neural networks, this could happen when too many neurons are in input or/and too many neurons are in a hidden layer, or with also too many hidden layers. The designing phase has to take into account this characteristic. In other words this happens when there are too many neurons in the network.

e. Critics about neural networks

To continue with other problems, different from overfitting:

The backpropagation process can suffer of one problem: There is no know configuration for this algorithm which enables it to always find the best solution, so a process of trial and error must be developed. This is generally done by redoing the descent a sufficient number of times, ensuring that the reached minimum is really the global one (although it cannot be proved).

This other intrinsic problem is that a neural network is a **black box** that is they can find patterns and relationship between inputs and output but they cannot exhibit them. So even if we can predict well the next value of a stock price, it is difficult to explicit the mechanism that yield to it.

2. Designing and neural paradigms

a. Variable selection as inputs

The first layer is the input representation, theses nodes take on the value of the input data. The selection of input variables is fundamental to accurately forecast the stock movements. It primarily depends on a clear understanding of the economic background of the stock price to forecast. The choice of input data influence the number of input neurons, so it have to be well chosen or reduced to avoid overfitting and to avoid non pertinent variables: this generally happens when a variable is highly correlated with another. The inputs can directly be the past values of a time series, or technical indicators, or fundamental indicators. The volume can be considered as a fundamental variable since it is an economic indicator (transactions amount).

b. Data preprocessing: Normalisation

Before the data is analysed by the neural network, it has to be pre-processed in order to increase the accuracy of the output and to facilitate the learning process of the neural network. This is a critical operation since neural networks are pattern matchers, thus the way data are represented directly influence their behaviour. Normalisation is useful to remove noisy data by creating more uniform distribution if needed. Indeed, such distribution tend to help the neural network forecasting process.

The input data in the majority of cases will thus require normalising, a process of standardising the possible numerical range that the input vector elements can take. The exception is for data that is already in a sense normalised, such as binary data, or data that is all composed from the same time series and presented in the same form. Even in this latter case, normalisation is advisable since the network training parameters can be tuned for a given range of input data, and can be carried over similar tasks. Given that normalisation is advisable, the method adopted requires consideration to the nature of the input data, there is no “correct” normalisation as such, the ultimate measure is whether the network is successful. The R function *monmlp.fit()* training a multi-layer perceptron standardizes the data. Along channel normalization was used: this method normalizes each input variable individually. We did a standard/mean scaling. While linear scaling strictly preserves the uniformity of the distribution, mean scaling helps in creating a more uniform distribution, which can be desired if most of the extreme values of the original distribution are considered to be noisy. And this is the case of financial time series.

Although the necessity of preprocessing data is not agreed by every researcher, there have been significant demonstrations showing that it can greatly improve the forecasting performance [27].

The target values of the output neuron also have to be normalized. In the same way as input data, the target has been standardised before training. For the target value in itself, different approaches can be made: we can use the price itself or the return etc.; in our case we tried both and as expected using the return is more appropriate.

c. Hidden Layer and Hidden Neurons

The second layer, and all subsequent layers, contains processing nodes, known as artificial neurons. Any layers between the input and output layers are called hidden layers. In general, network designs may contain typically one or two hidden layers with many neurons per layer. The hidden layers are composed by the set of all neurons between the input and the output neurons. The number of hidden layer that should be used cannot be clearly defined as it really depends on the amount of input neurons and the properties of the data. Formulas were developed which tried to take into account those parameters, but due to the nature of the stock movements, it cannot be predicted easily. However, it is commonly admitted that one or two hidden layers are enough, and that increasing the amount of those layers also increases the danger of overfitting and the computation time.

The most common technique used today to determine the appropriate number of hidden neuron to include in the model is experimentation. It is a part of the training phase of the neural network development and might require a lot of computations. The only rule to keep in mind while selecting the most appropriate number of hidden neurons is to always select the network that performs best on the testing set with the least number of hidden neurons. This was the way used to choose the number of hidden layers and the number of hidden neurons.

In our case the second hidden layer was usefulness. The number of hidden neurons was in most cases inferior to the number of input neurons.

d. *Implementation in R*

To implement the neural networks in our study we used the CRAN package *monmlp* which implements the multi-layer perceptron.

Step 1:

Choice of the input variables, the number of hidden layer, the number of hidden neurons.

Like we said before regarding the target value (next price), the cost function (MSE) and the activation functions (hyperbolic tangent function for hidden neurons and linear function for the output neuron), it is already determined and their values were constant during the study.

Step 2:

The function *monmlp.fit()* was used to train the network, so to determine with the error backpropagation and the training set, the different weights of the network.

Step 3:

We used the function *monmlp.predict()* to do the validation phase with the validation set that is one-step-ahead predictions. We tried different neural network structure (see step 1) and compare them with forecasting performances.

We selected the one which outperformed the others. To avoid overfitting the structure of hidden layers did not have to be so complex.

3. First NN with past values as inputs

The first neural network we tried is in continuity with our previous models. Indeed we used as inputs five past values of our time series to predict the next successive value. This type of neural network can also be viewed like a generalization of an autoregressive model since it is like a non-linear autoregressive model trying to find a non-linear function between past values and future values. We chose 5 inputs because like we have seen with the linear models a coefficient of four for the autoregressive parts was sufficiently enough. Furthermore to be able to compare this model with the second neural network we have to choose the same number of input.

Like we said before, we actually used the returns to proceed in our neural network for both inputs and output.

This model is quite similar to the STAR in its definitions: it is a non-linear autoregressive model.

4. Second NN with inputs capturing the market dynamics

The second one begin to exploit the great flexibility of neural networks since inputs can be chosen as various as possible.

To capture the market dynamics, that is to implement the real stock price mechanism, a good way to choose neural network inputs is to use the open, high, low and close prices and the volume of an instant t.

In our study we used intraday data at a frequency equals to 15 minutes. The open price is the price at t, the close price is the price at t+15 minutes (if we consider t in minutes), the high price is the highest price of the period between t and t+15 and at the contrary the low price is the lowest price of the period between t and t+15. The volume is the transactions amount at the time during the given period.

So we have five inputs as before. We used the same pre-processing as the previous NN since we found that we obtained better forecasting results using returns instead of the price themselves (only for closing prices and output).

5. Other models studied

We also tried different neural networks with diverse structures. Nevertheless the models we will briefly describe here were not performing so we did not describe the results in the following section.

We tried to implement as inputs technical indicators. We put around 60 technical indicators in a neural network. Technical indicators are computing using the volume and the low, high, open, close prices. There can be indicators of trends, of volume, of volatility or momentum (empirically observed tendency for rising asset prices to rise further). the most common indicators are the relative strength index RSI, (A technical momentum indicator that compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset), the Money Flow Index (MFI): this one measures the strength of money in and out of a security, or the Stochastic Oscillator (SO): This function compares a security's closing price to its price range over a given time period.

To avoid overfitting and computing time we had to reduce the number of inputs. This is also necessary to only select the pertinent variables (eliminate those which are correlated to others, the ones which thus do not have an impact on the target value)

There were two different approaches to do this:

- *The sensitivity analysis*: The sensitivity analysis is the process which determines whether an input variable influence the output of the neural network or not. There are several ways to find this. The most common approach is to run the neural network with and without each input variable, and to check the variations of the input. If there are no noticeable changes, it surely means that the input variable can be omitted. This generally happens when a variable is highly correlated with another. Because those variables where chosen for being related to the current stock to forecast, such a situation is likely to happen. The other approach we actually used is to analyse the correlation matrix of the inputs and to eliminate those which were correlated to others (with a correlation coefficient > abs (0.5)). So we select with dichotomy the pertinent inputs. This work was necessary since we can see that many technical indicators are correlated and this work could lead us to reduce the inputs from 60 to around 15.
- *The component principal analysis*: the PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a

smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. So we chose as totally uncorrelated inputs in the neural networks the first PC which explained at least 90% of the variance.

To conclude this part and a little anticipate the results: neural networks are good forecasters and outperformed the previous models. Nevertheless, they are the ones which demand the most computing time since with the problems we talked about earlier:

- Problem of initial weights: several computations have to be made.
- Problem of alchemy with the hidden layers and hidden neurons: time to find the best model.
- Problem of the backpropagation algorithm and the possible evidence of local minima: computations have to be redone.

With this we have to add the intrinsic computing time linked with the determination of weights.

VII. Out-of-sample forecasting performance results

Last but not least, after presenting some generalities in this field, our methodologies and the models we used, here are the fruits of our work: the results after the R computing. We will present the forecasting performances of the different models and discuss them in a global view after it.

1. Presentation of the results

a. Considerations

Here are following the different tables for the forecasting results based on statistical accuracy (Theil's U) and forecasting results based on an economical accuracy (HIT) for the different models used in our study (linear, non-linear and machine learning models) for the ten companies.

In **bold** is the best model chosen for each company which can be easily chosen if it has the best HIT and the best Theil's U otherwise you have to consider a trade-off between a best HIT and a best Theil's U depending on the case in consideration. Nevertheless the Theil's U was preferred to the HIT rate since it is more accurate (HIT rate includes a part of "chance" of the model because there is probability of 0.5 to be right). The HIT rate was used to prefer a model with a really great HIT and a lower - but not bad - Theil's U than the one of the best model (Theil's U speaking). Actually in most cases the one which had the best Theil's U indicator was the one with the best HIT rate either or at least one of the best and its HIT rate was also by the way superior to the 50% value and thus to the one from the random walk.

We can also see in the tables, statistical accuracy indicators like RMSE, MAE and MAPE as information.

We added, to the performance indicators' tables, one graph, the evolution of Theil's U for the diverse models to see well how they outperform the random walk for each company.

So for each company stock, we present the forecasting performances of the random walk (naïve prediction), the ARIMA, ARFIMA, Holts Winters, ARMA-GARCH, STAR, Neural Network 1, Neural Network 2 by a numerical table and by a visual indication graph.

Some companies do not have forecasting indicators for the STAR model since the linearity test was accepted by the procedure (4 companies out of 10).

Finally we added two graphs as a conclusion; the first one is the evolution of the mean of Theil's U for a given model, mean computing for the ten companies and the other one is the HIT rate medium computing in the same way.

Only for the first company (otherwise it could have not-usefully over-increased the number of pages), we added a table with the coefficients of the best model (ARIMA), a qq-plot of target prices vs. predicted prices, a plot of the evolution of the target values and of the predicted values for the ARIMA (for stocks) and for the NN1 (for returns).

b. *Results: tables and graphs*

Table 6: Forecasting performances for company 1

Company[1]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.125593	0.093450	0.248398	1.000000	50
ARIMA(2,1,1)	**0.119701**	**0.090143**	**0.239580**	**0.950617**	**62**
ARFIMA	0.121618	0.091312	0.242693	0.967869	56
HOLT-WINTERS	0.121590	0.091391	0.242896	0.967660	54
STAR	-----	-----	-----	-----	-----
ARMA(2,2)-GARCH(1,1)	0.120390	0.090549	0.240658	0.955995	63
Neural Network 1 (5,2,1)	0.121111	0.092019	0.244581	0.961853	60
Neural Network 2 (5,4,1)	0.120630	0.091516	0.243192	0.960240	58

Analysis: For the first company, we had a great predictability since the Theil's U was under 0.97 for each model and every model outperforms the 50% value for the HIT rate. The linearity test was accepted and the ARIMA model justifies it since it is the best model for this stock with a good forecasting (U<0.96). We can see on the next table the coefficients for this best model and their p-value: it is well accepted. The ARMA-GARCH is at the second place.

Table 7: Coefficients of the ARIMA[1]: the best model

ARIMA(2,1,1)	**Estimate**	**Standard Error**	**t value**	**p value**
Ar1	-1.05242	0.03843	-27.39	< 2e-16
Ar2	-0.12672	0.02509	-5.05	4.42e-07
Ma1	0.94856	0.03028	31.32	< 2e-16
AIC	-1853.28			

The next graph shows a QQ-plot between the target values and the predicted values for this ARIMA(2,1). Even if this graph is not accurate for a prediction performance it can easily show if the predictions are biased. In our case the results can be promising.

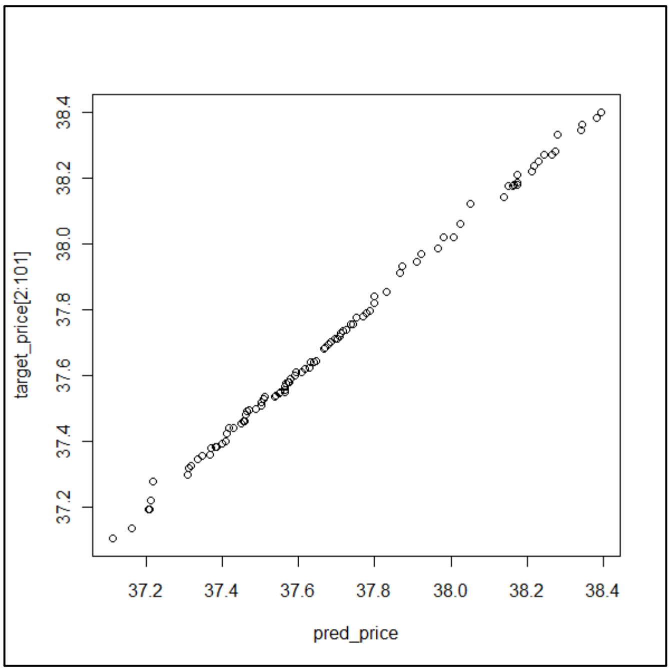


Figure 14: QQplot of target prices vs. predicted prices [1]

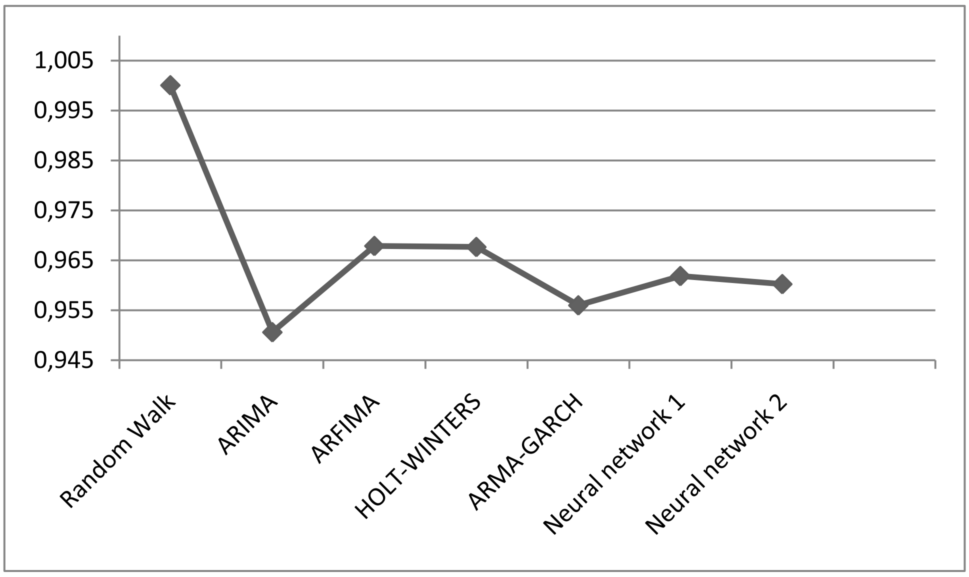


Figure 15: Theil's U [1]

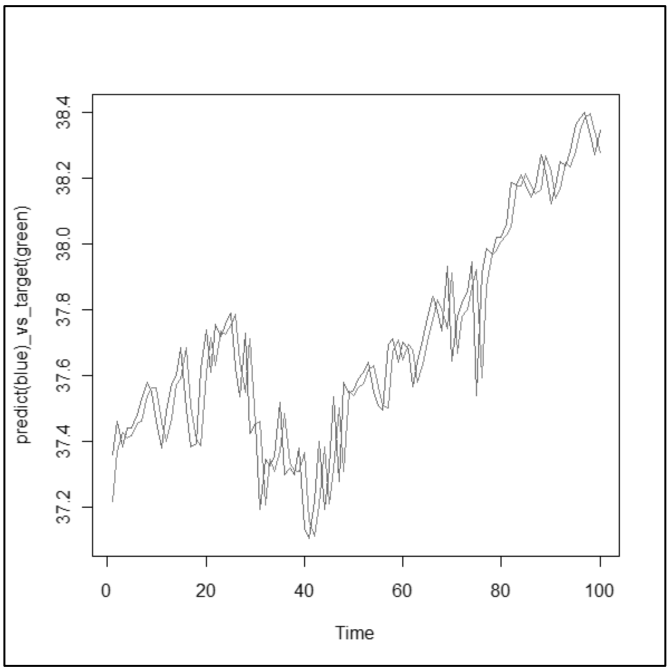


Figure 16 & 17: Evolution of the target values and of the predicted values for stocks of the ARIMA[1] (up) and for the returns of NN1 (down)

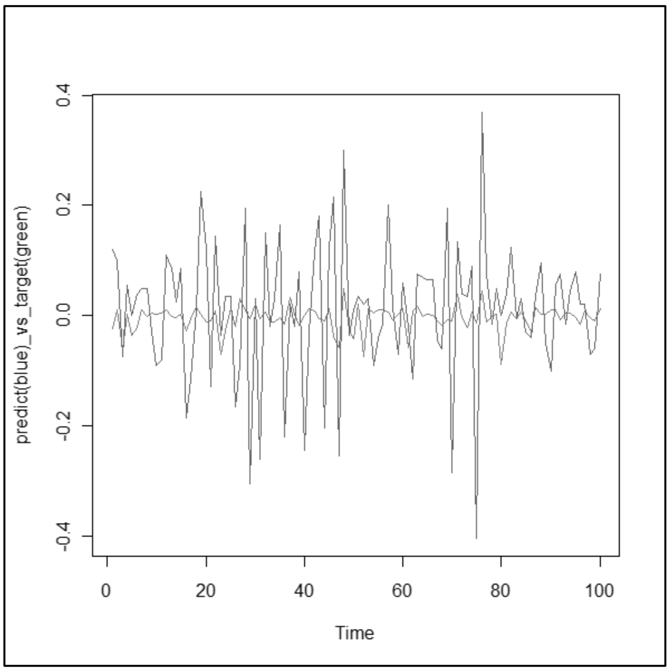


Table 8: Forecasting performances for company 2

Company[2]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.050032	0.037550	0.280275	1.000000	50
ARIMA(4,1,2)	0.049447	0.036829	0.274880	0.990597	55
ARFIMA	0.049932	0.037379	0.278986	0.999251	50
HOLT-WINTERS	0.049853	0.037378	0.278997	0.997093	52
STAR	0.049705	0.037527	0.280143	0.992543	50
ARMA(1,1)-GARCH(1,1)	0.049834	0.037456	0.279622	0.995326	52
Neural Network 1 (5,1,1)	0.049978	0.037511	0.279964	0.999912	51
Neural Network 2 (5,2,1)	**0.049040**	**0.037757**	**0.281920**	**0.979254**	**53**

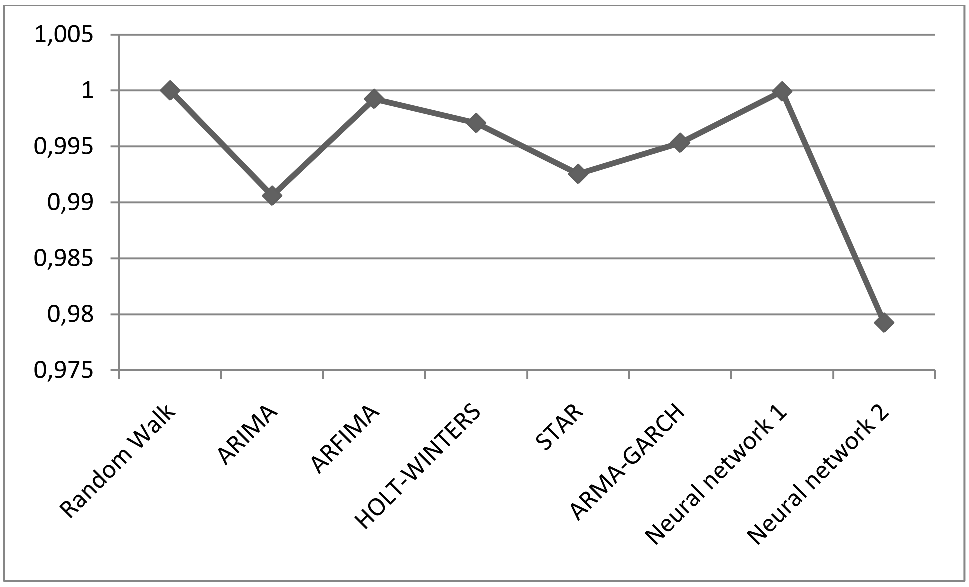


Figure 18: Theil's U [2]

Analysis: for this company the neural network 2 outperformed the forecasting performances against the other models. The ARIMA takes the second place. For this first non-linear time series, the STAR takes the next place.

Table 9: Forecasting performances for company 3

Company[3]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.179359	0.144400	0.181310	1.000000	50
ARIMA(3,1,2)	0.179007	0.143690	0.180411	0.996499	50
ARFIMA	0.179289	0.144378	0.181286	0.999532	52
HOLT-WINTERS	0.179621	0.144311	0.181196	1.001451	53
STAR	-----	-----	-----	-----	-----
ARMA(2,2)-GARCH(1,1)	0.179155	0.143585	0.180284	0.996961	53
Neural Network 1 (5,3,1)	0.180572	0.144567	0.181621	0.996563	56
Neural Network 2 (5,5,1)	**0.175789**	**0.141369**	**0.177492**	**0.980513**	**54**

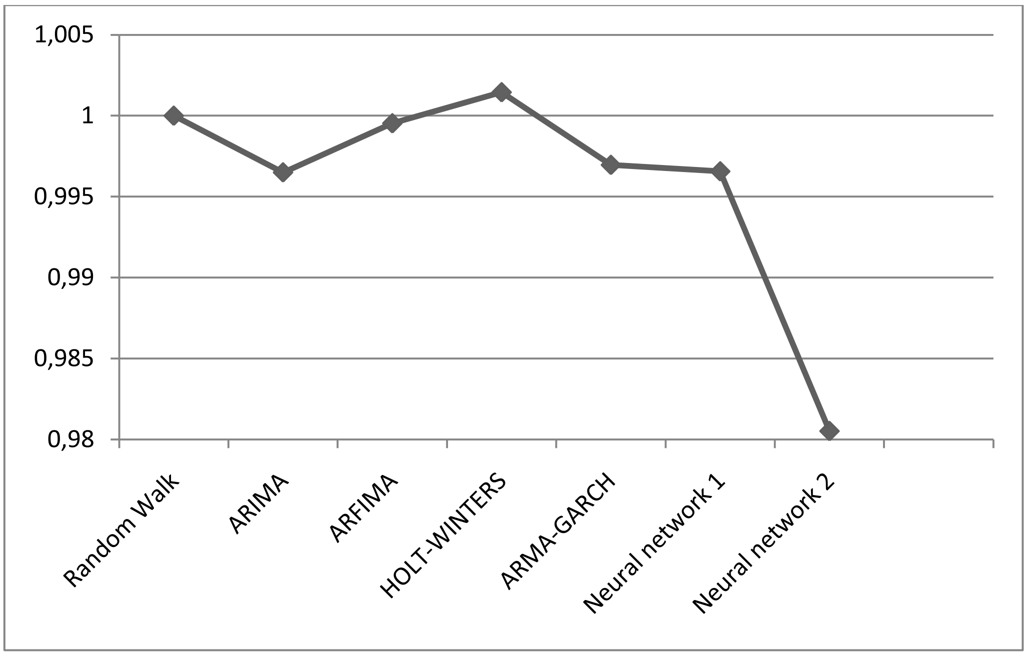


Figure 19: Theil's U [3]

Analysis: for this stock it is the first time that a model does not outperform the random walk (see the Holt-Winters model). As previously the NN2 outperformed the models followed by the NN1.

Table 10: Forecasting performances for company 4

Company[4]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.131225	0.094650	0.196255	1.000000	50
ARIMA(2,1,1)	0.131012	0.094740	0.196455	0.999020	55
ARFIMA	0.130972	0.094690	0.196361	0.997488	48
HOLT-WINTERS	0.131207	0.094648	0.196253	0.999845	47
STAR	0.131670	0.095349	0.197720	1.003734	48
ARMA(2,2)-GARCH(1,1)	0.130625	0.094406	0.195774	0.996397	53
Neural Network 1 (5,3,1)	0.129785	0.093536	0.193965	0.989751	56
Neural Network 2 (5,3,1)	**0.122884**	**0.085059**	**0.187146**	**0.974100**	**54**

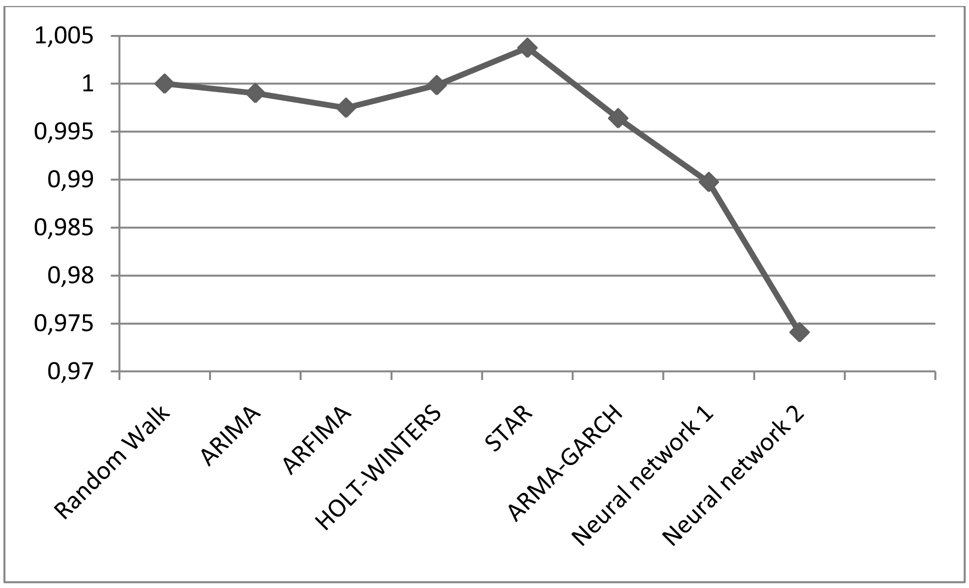


Figure 20: Theil's U [4]

Analysis: even if the non-linearity test was accepted the STAR did not outperform the random walk. The two neural networks were the best with at the first place again the NN2. We can see that three non-linear models are at the three first places as expected with the white test.

Table 11: Forecasting performances for company 5

Company[5]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.006682	0.005050	0.217277	1.000000	50
ARIMA(3,2,2)	0.006674	0.005084	0.218730	1.000459	51
ARFIMA	0.006689	0.005087	0.218866	1.002561	52
HOLT-WINTERS	0.006698	0.005080	0.218578	1.002851	44
STAR	0.006695	0.005061	0.217754	1.003790	53
ARMA(1,0)-GARCH(1,1)	0.006693	0.005089	0.218964	1.003103	53
Neural Network 1 (5,3,1)	0.006600	0.005064	0.217870	0.988938	52
Neural Network 2 (5,3,1)	**0.006487**	**0.005006**	**0.215386**	**0.971391**	**53**

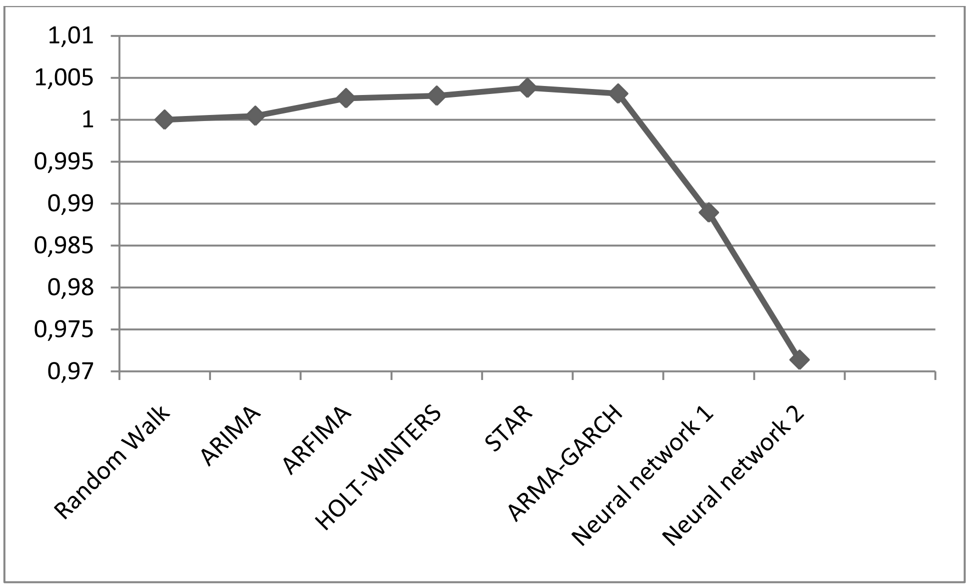


Figure 21: Theil’s U [5]

Analysis: This company was quite difficult to forecast. Actually, like we can see, only two models outperformed the random walk: the two neural networks with the same ranking as before.

Table 12: Forecasting performances for company 6

Company[6]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk - naive	0.081084	0.059950	0.146206	1.000000	50
ARIMA(1,1,1)	0.081053	0.059933	0.146170	0.999696	51
ARFIMA	0.080647	0.059679	0.145561	0.994482	53
HOLT-WINTERS	0.080998	0.059923	0.146143	0.998954	50
STAR	0.080884	0.060529	0.147634	0.997501	48
ARMA(1,0)-GARCH(1,1)	0.080519	0.059979	0.146291	0.993011	47
Neural Network 1 (5,3,1)	**0.079724**	**0.058445**	**0.142551**	**0.983063**	**59**
Neural Network 2 (5,5,1)	0.080574	0.059641	0.145477	0.993115	47

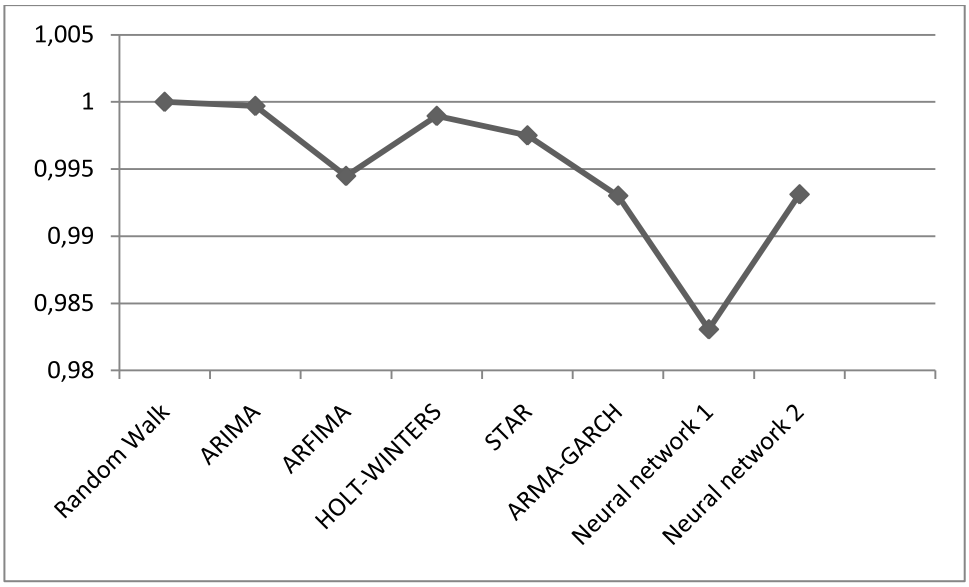


Figure 22: Theil's U [6]

Analysis: at this time the neural network 1 outperformed the forecasting performances with a great HIT rate quite superior to the others. For this non-linear time series (see non-linearity test) the three last non-linear models outperformed the linear models.

Table 13: Forecasting performances for company 7

Company[7]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk	0.181562	0.126900	0.231841	1.000000	50
ARIMA(3,2,2)	0.180941	0.126083	0.230336	0.996485	49
ARFIMA	0.180848	0.125876	0.229960	0.995607	52
HOLT-WINTERS	0.181350	0.126618	0.231309	0.998610	46
STAR	-----	-----	-----	-----	-----
ARMA(1,1)-GARCH(1,1)	0.180974	0.125615	0.229484	0.995995	51
Neural Network 1 (5,3,1)	0.180079	0.125264	0.228841	0.990505	51
Neural Network 2 (5,7,1)	**0.176916**	**0.127658**	**0.233171**	**0.973871**	**53**

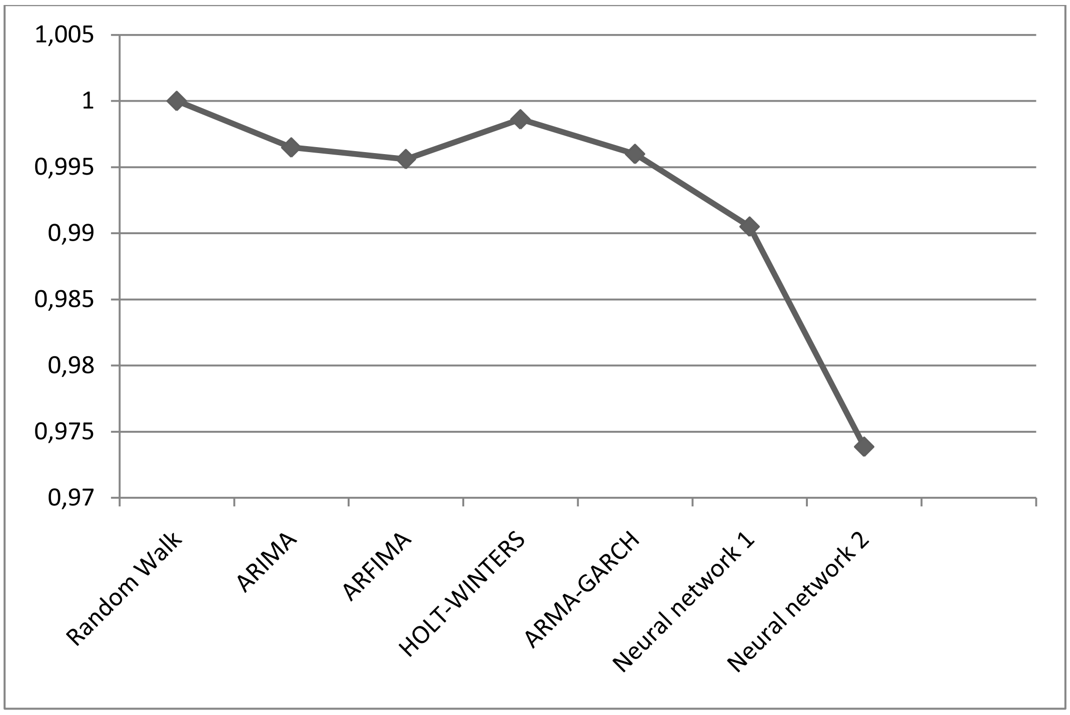


Figure 23: Theil's U [7]

Analysis: As usual now the neural networks outperformed the forecasting accuracy with the second NN at the first place.

Table 14: Forecasting performances for company 8

Company[8]	**RMSE**	**MAE**	**MAPE**	**Theil's U**	**HIT**
Random Walk	0.061607	0.046700	0.141885	1.000000	50
ARIMA(1,1,0)	0.061586	0.046444	0.141108	0.999274	55
ARFIMA	0.061465	0.046586	0.141533	0.997100	49
HOLT-WINTERS	0.061590	0.046435	0.141080	0.999340	55
STAR	0.061966	0.046438	0.141094	1.005543	48
ARMA(2,2)-GARCH(1,1)	0.061752	0.046456	0.141150	1.001997	54
Neural Network 1 (5,3,1)	0.061532	0.047197	0.143361	0.993686	50
Neural Network 2 (5,3,1)	**0.058446**	**0.045398**	**0.137879**	**0.957580**	**56**

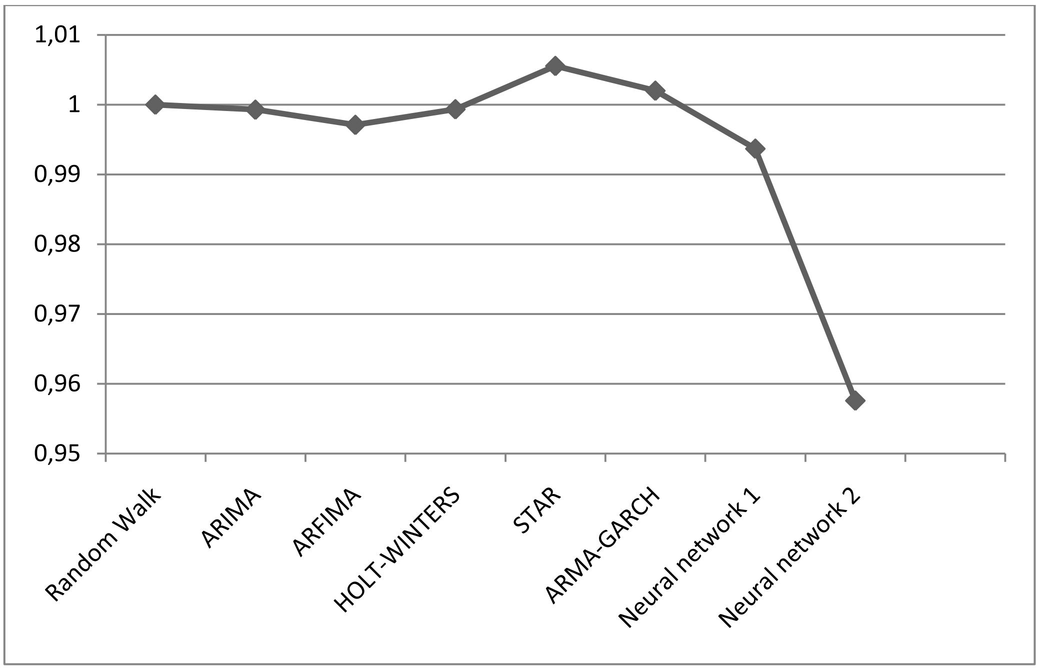


Figure 24: Theil's U [8]

Analysis: this graph is similar to the previous ones since neural networks outperformed the other models. The NN2 was in this case a really good forecaster with a Theil's U under the 0.96 and a HIT rate of 56.

Table 15: Forecasting performances for company 9

Company[9]	RMSE	MAE	MAPE	Theil's U	HIT
Random Walk	0.083144	0.064800	0.209425	1.000000	50
ARIMA(2,1,1)	**0.082468**	**0.063497**	**0.205177**	**0.991312**	**57**
ARFIMA	0.082628	0.063665	0.205763	0.993530	58
HOLT-WINTERS	0.083031	0.063972	0.206733	0.998785	55
STAR	0.083490	0.064369	0.207977	1.004228	56
ARMA(2,2)-GARCH(1,1)	0.082729	0.064116	0.207176	0.994469	53
Neural Network 1 (5,3,1)	0.082527	0.063342	0.204690	0.991985	58
Neural Network 2 (5,3,1)	0.082772	0.064259	0.207695	0.995299	57

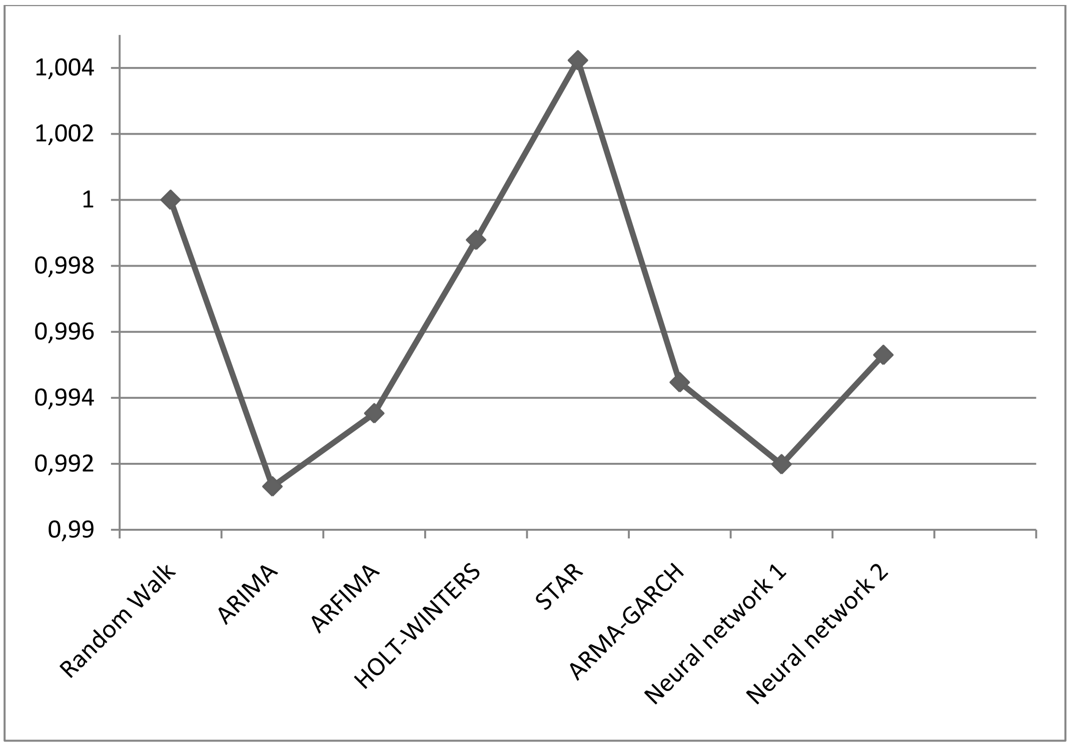


Figure 25: Theil's U [9]

Analysis: Quite surprising for this non-linear time series, the ARIMA was the best. Near-followed by the NN1.

Table 16: Forecasting performances for company 10

Company[10]	RMSE	MAE	MAPE	Theil's U	HIT
Random Walk	0.055486	0.041150	0.259050	1.000000	50
ARIMA(1,2,1)	0.055372	0.040671	0.256022	0.998139	58
ARFIMA	0.055372	0.040670	0.256015	0.998148	58
HOLT-WINTERS	0.055640	0.040878	0.257352	1.002538	53
STAR	-----	-----	-----	-----	-----
ARMA(1,0)-GARCH(1,1)	0.055441	0.040709	0.256265	0.999217	56
Neural Network 1 (5,3,1)	**0.053529**	**0.039217**	**0.246878**	**0.967185**	**67**
Neural Network 2 (5,6,1)	0.054786	0.041555	0.261700	0.986499	50

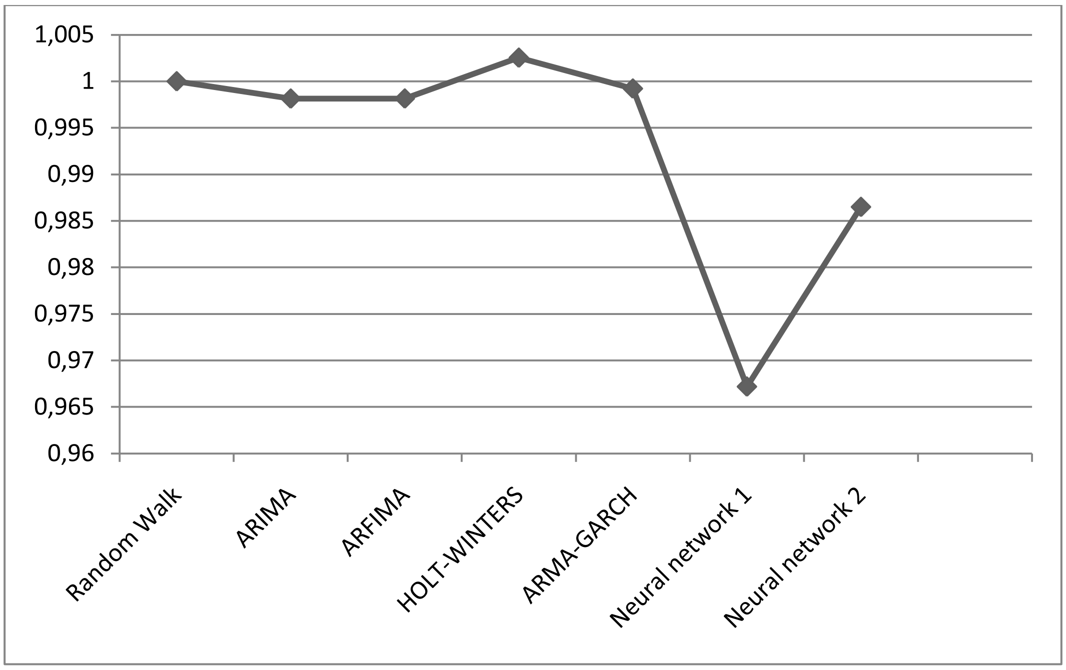


Figure 26: Theil's U [10]

Analysis: Maybe our best hope for next researches, this company stock was very well forecasted by the NN1 with a good Theil's U and a great HIT rate equalling to 67%.

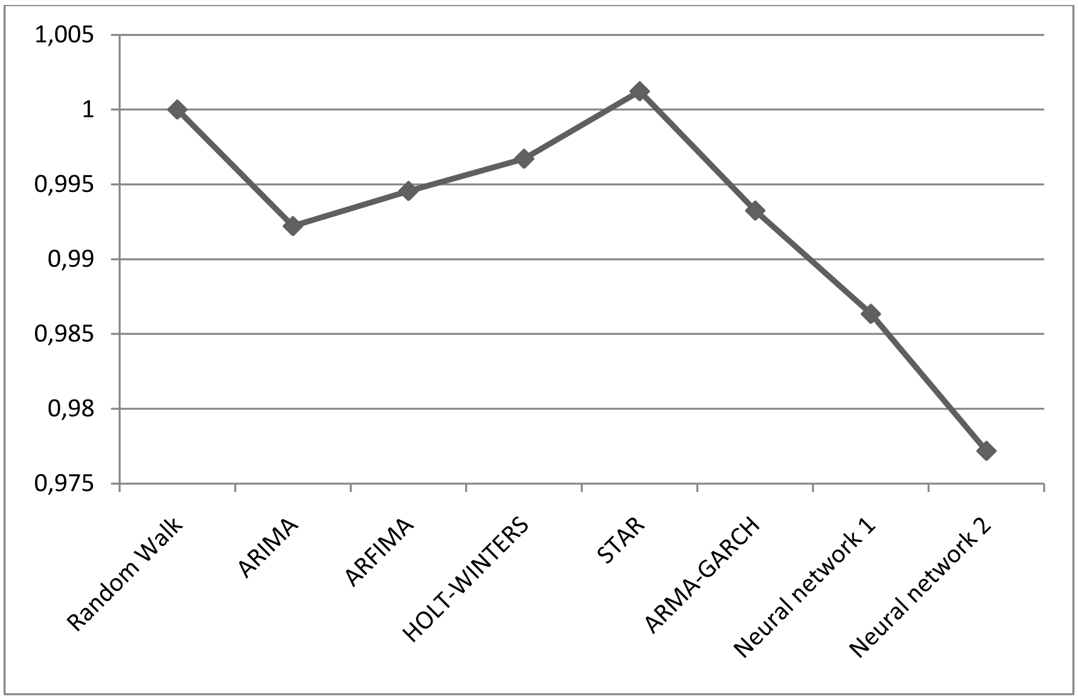


Figure 27: Mean of Theil's U indicators

This figure shows that neural networks outperformed the other model with the statistical indicator. At the first place far from the others the NN2 followed by the NN1. The ARIMA took the third place.

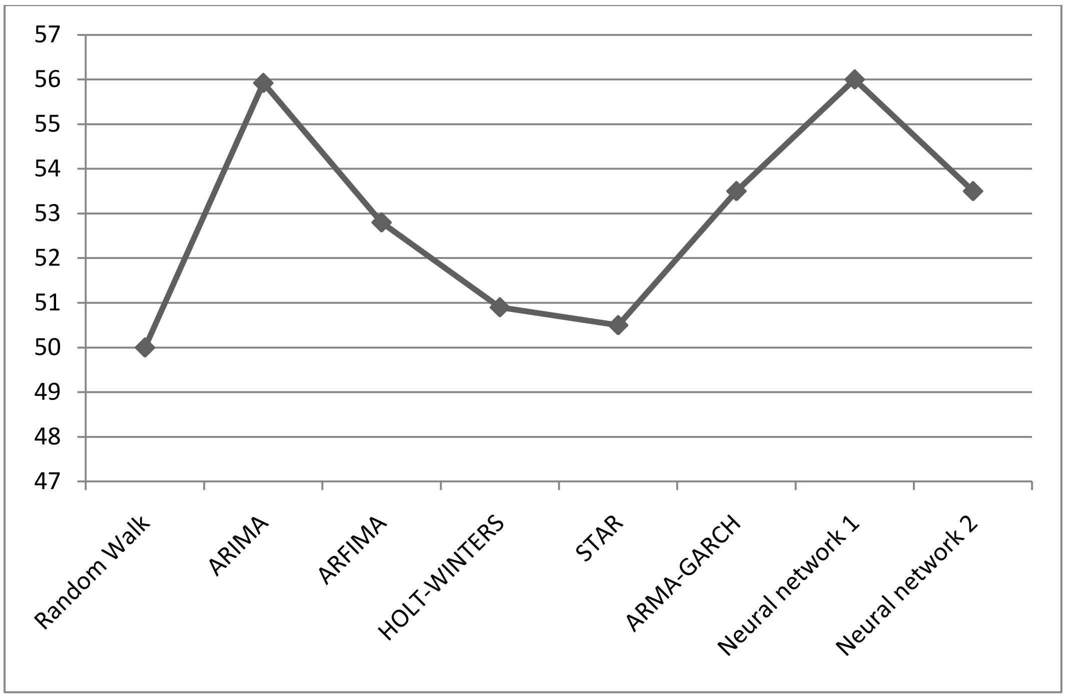


Figure 28: Mean of HIT

As far as the HIT economic indicator is concerned, the ARIMA and NN1 are at the first places. We can see that even if the NN2 outperformed the models with the statistical indicator, the NN1 was the best “economic” model with an HIT rate of 56%.

Considering a trade-off between these two forecasting performance indicators, the NN1 seems to be the best model of our study, since it outperforms the NN2 for the economic indicator (56 against 53.5) and it is as the second place with the Theil’s U indicator.

Anyway, the both neural networks showed that for each indicator they outperformed the linear models and the non-linear models. It also shows the great forecasting capacity of machine learning.

CONCLUSION AND PERSPECTIVES

We have seen that high-frequency financial time series can be forecasted since we obtained good results (HIT>50% and around 60% for the best models) and that the random walk and the naïve prediction were beaten (Theil's U < 1). Neural Networks showed their great predictability power since they outperformed the forecasting performances regarding the other models.

However, to test correctly the EMH, the forecasting should be done in real-time since in real markets investors' current and future forecast of payoffs affect their current and future trades which in turns affect returns, i.e., there is a feedback mechanism which has not been considered. Furthermore, if investors start to apply this forecasting methodology the temporary forecasting ability that exists according to the EMH will quickly disappear and, hence, the EMH will hold. In this sense, by applying more sophisticated trading strategies the financial markets will become more efficient.

Moreover, the weak form of the EMH does not hold in this financial market and there are trading opportunities like we have seen but the gain is maybe too small when compared with the transaction costs to take full advantage. Indeed no transaction costs were taking into account.

So to go on this study, trading strategies must be added with absolutely taking into account transaction costs.

APPENDIX

1. Statistical descriptions

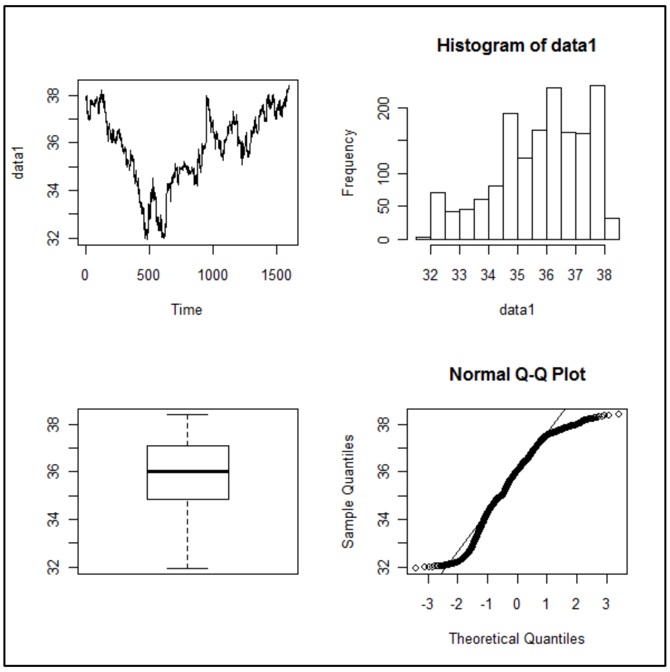
In this first appendix, we put some graphs regarding our data. For each company we can first find the plot of the stock, the histogram of the stock, its box plot and its qq-plot.

We see then the diagrams of autocorrelation and partial autocorrelation of the stock and the return for each company too.

Finally there are the plots of the returns themselves and their statistical graphs as for stocks (histogram, qq-plot and boxplot)

All these graphics were done with the whole dataset.

Theses graphics were more described and analysed in section II.



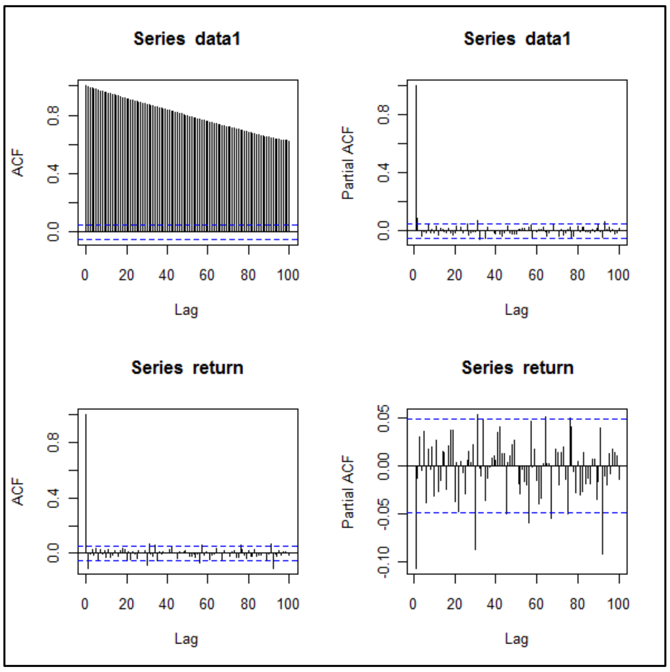
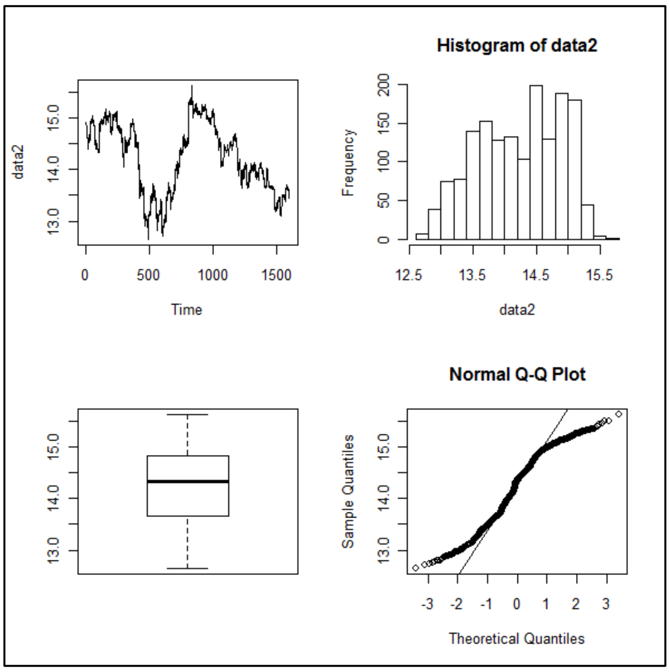


Figure 29 & 30: Statistical graphs for [1]



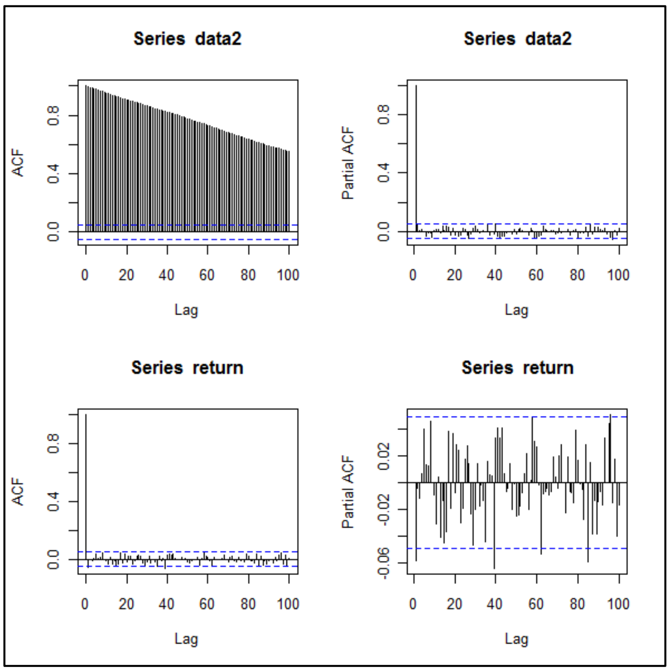
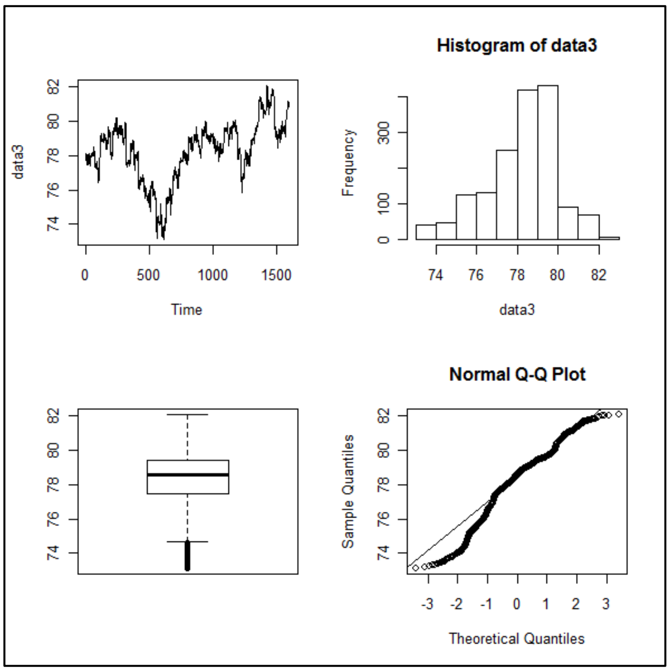


Figure 31 & 32: Statistical graphs for [2]



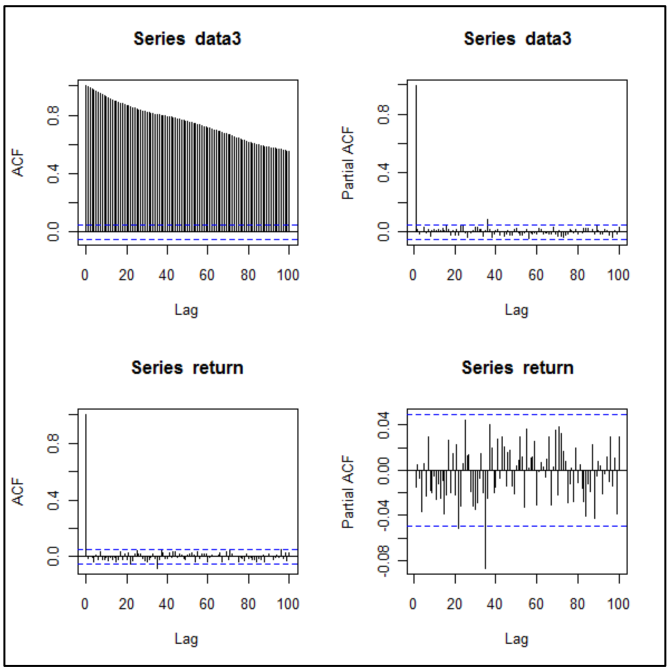
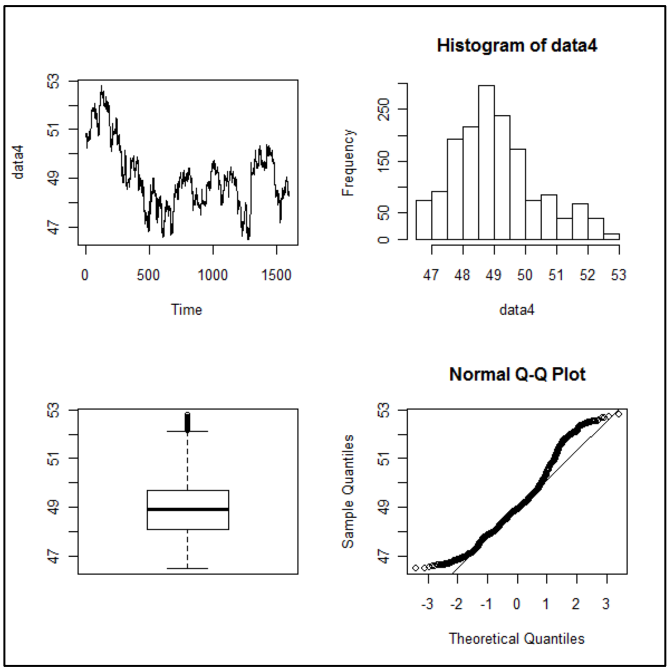


Figure 33 & 34: Statistical graphs for [3]



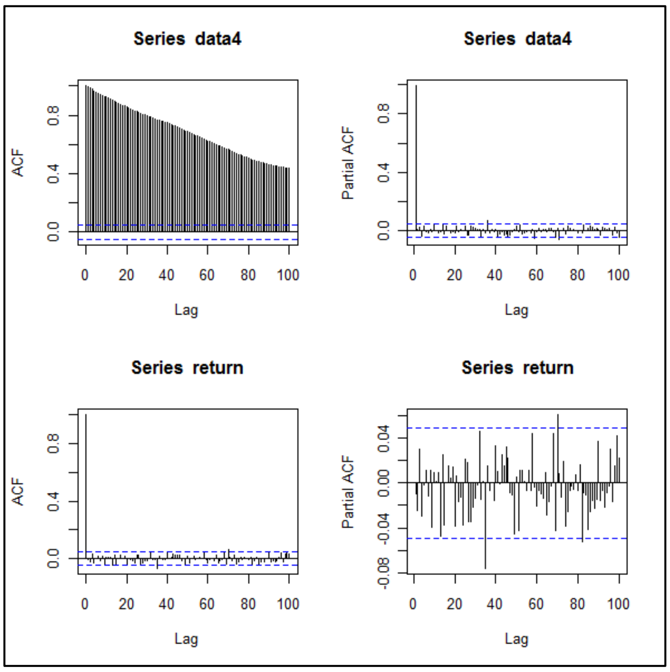
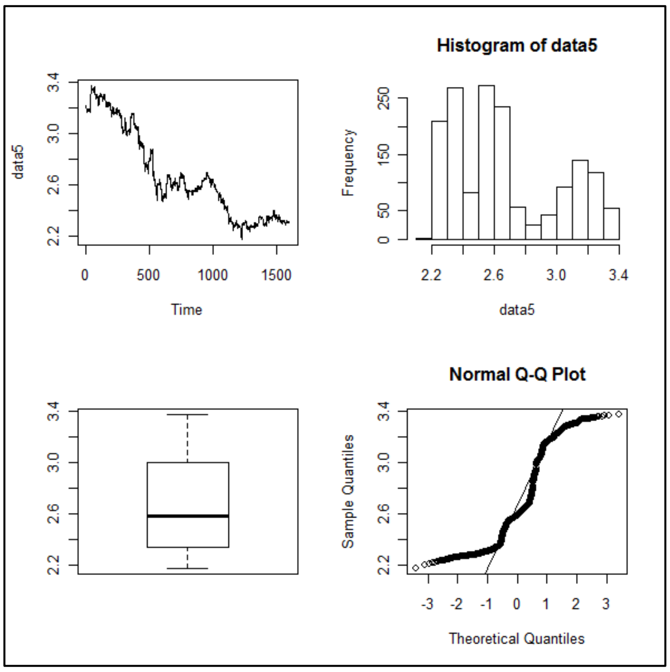


Figure 35 & 36: Statistical graphs for [4]



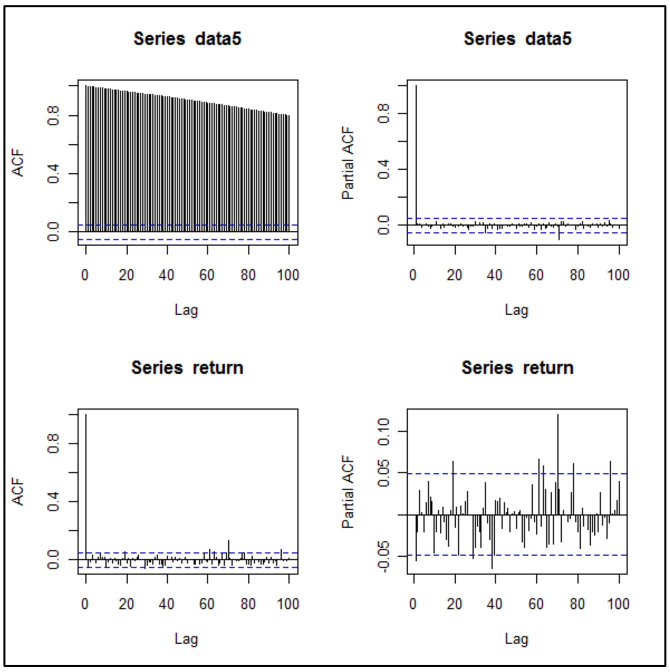
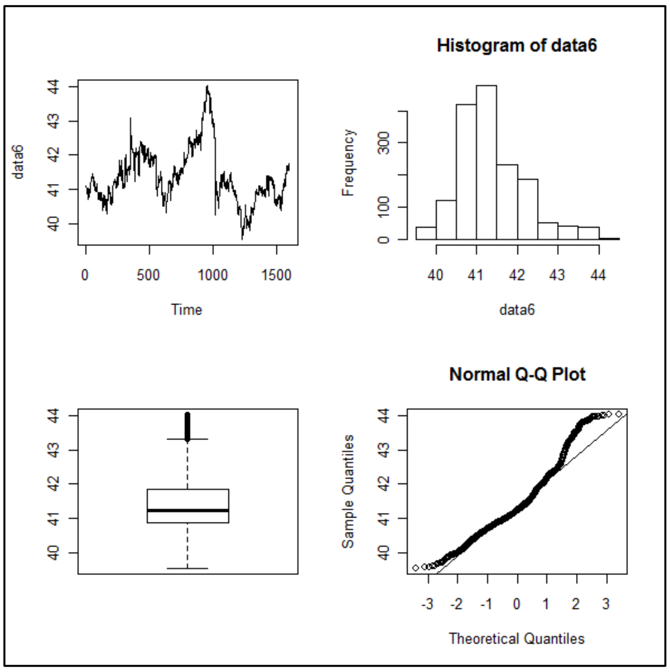


Figure 37 & 38: Statistical graphs for [5]



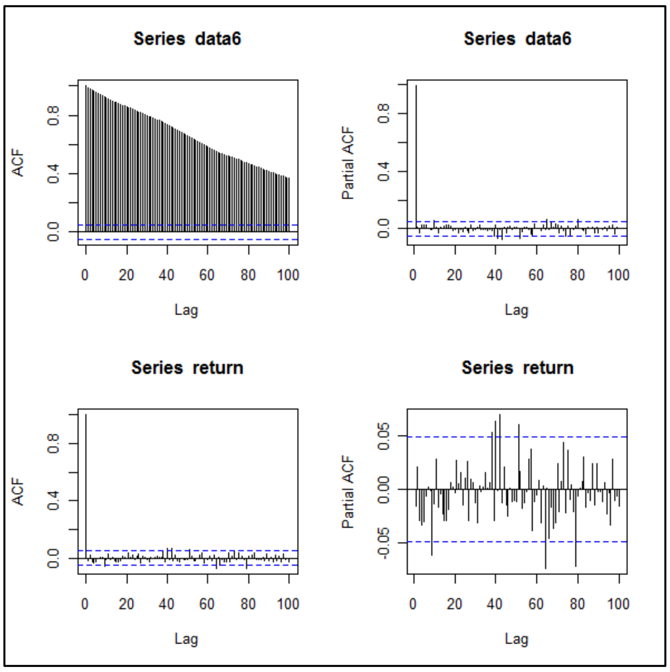
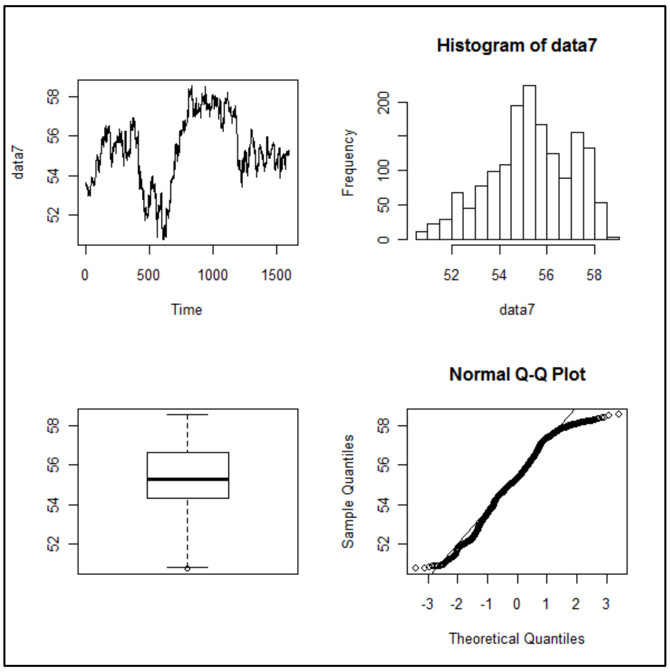


Figure 39 & 40: Statistical graphs for [6]



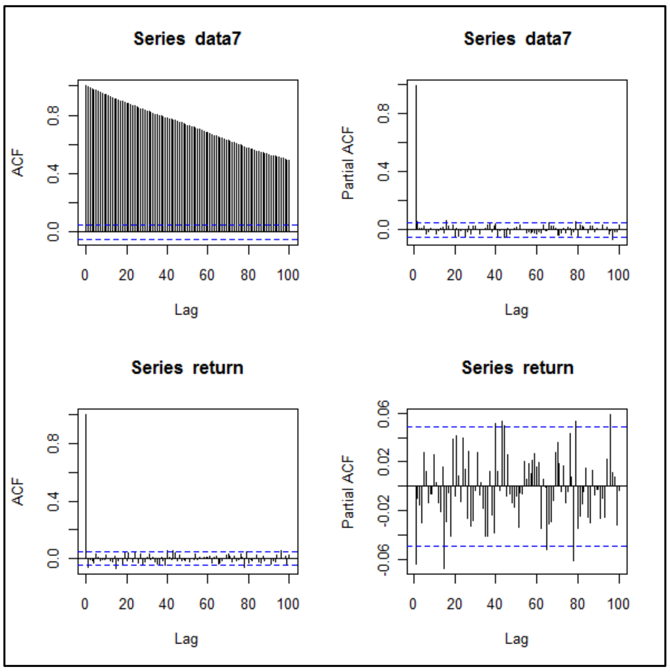
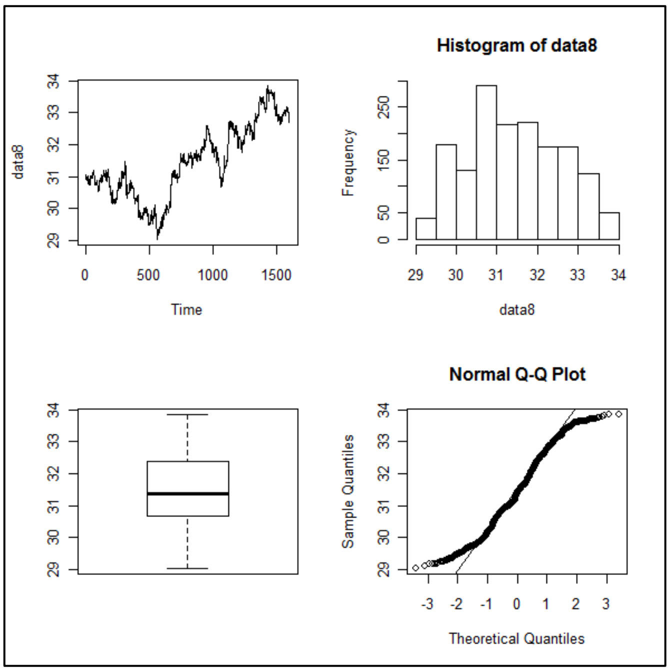


Figure 41 & 42: Statistical graphs for [7]



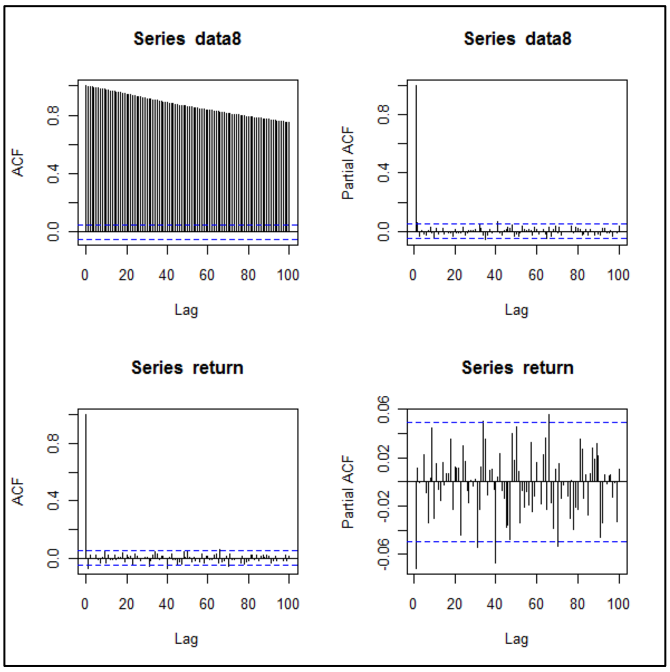
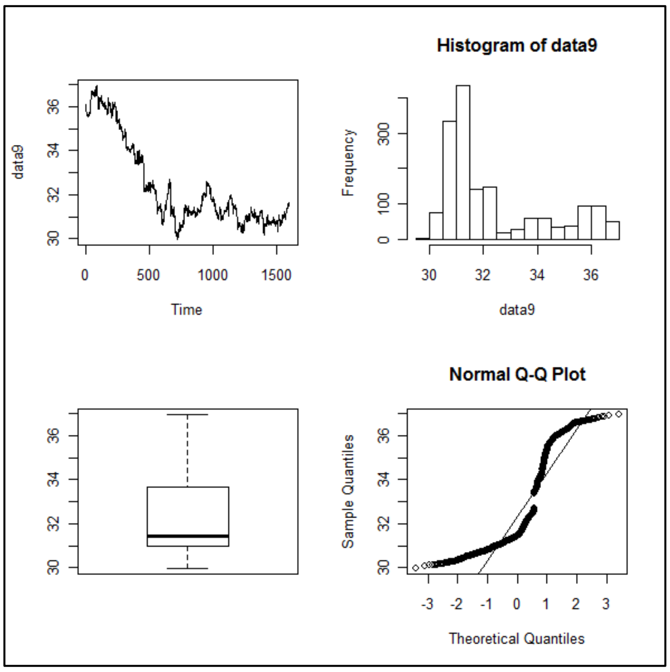


Figure 43 & 44: Statistical graphs for [8]



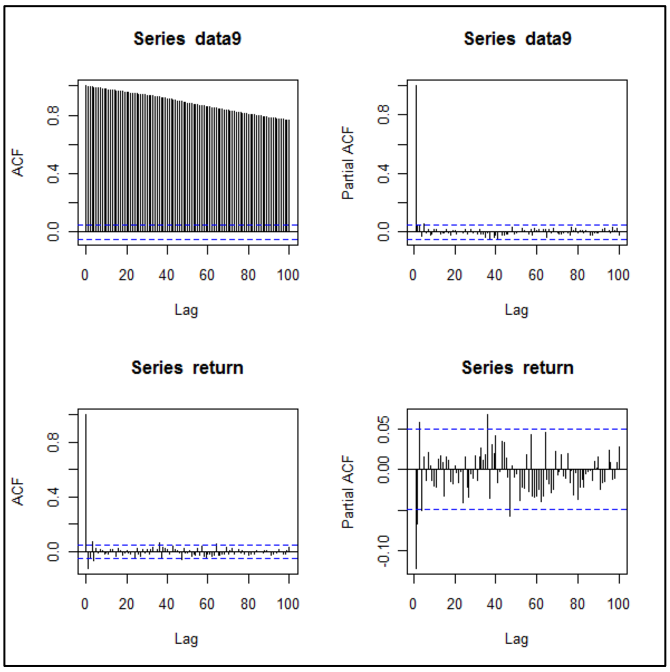
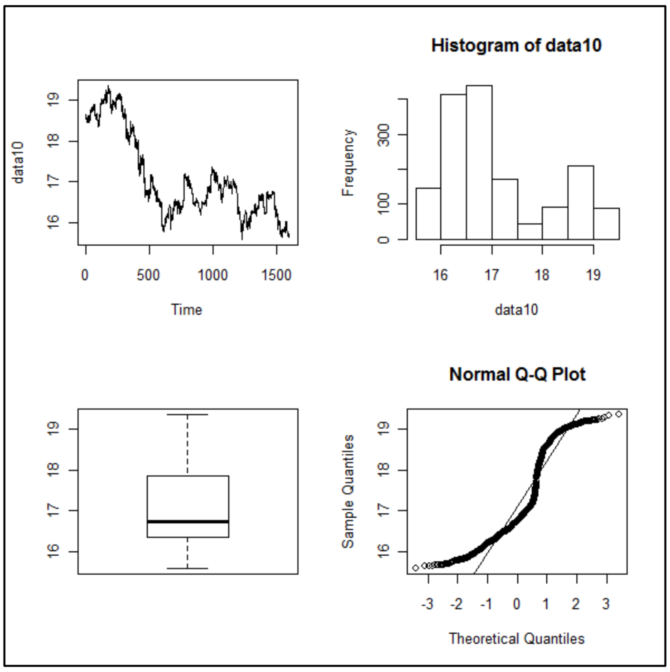


Figure 45 & 46: Statistical graphs for [9]



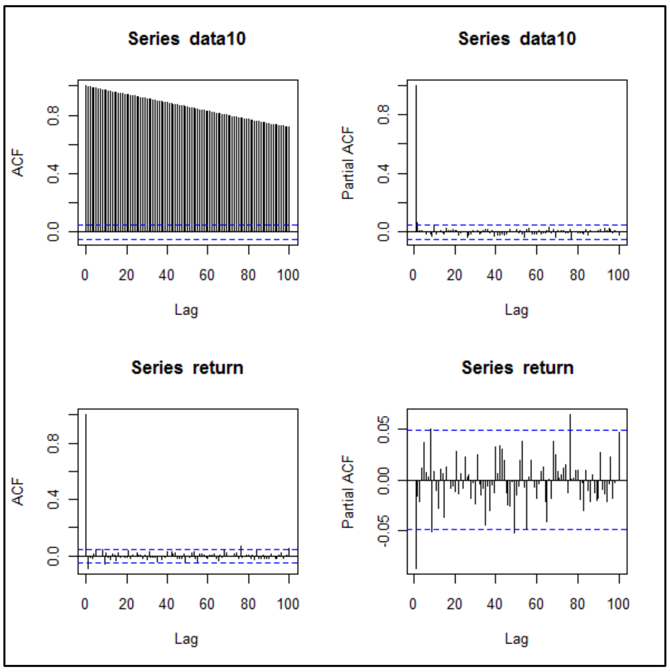
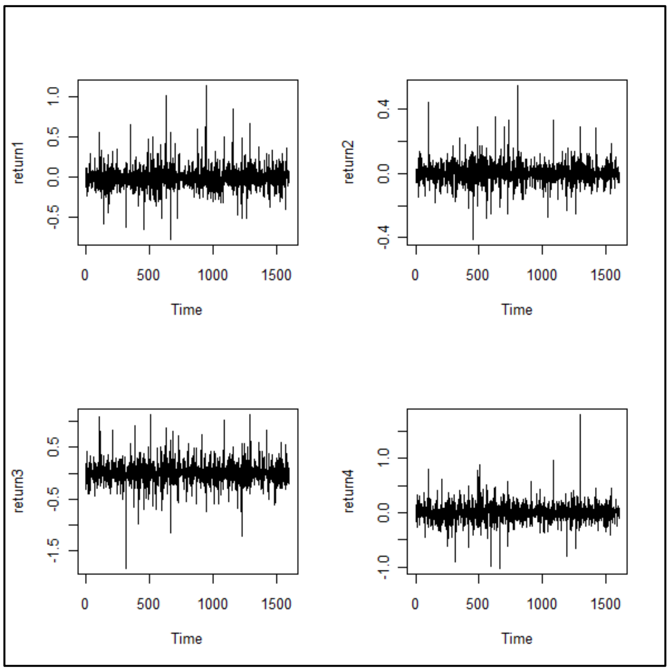


Figure 47 & 48: Statistical graphs for [10]



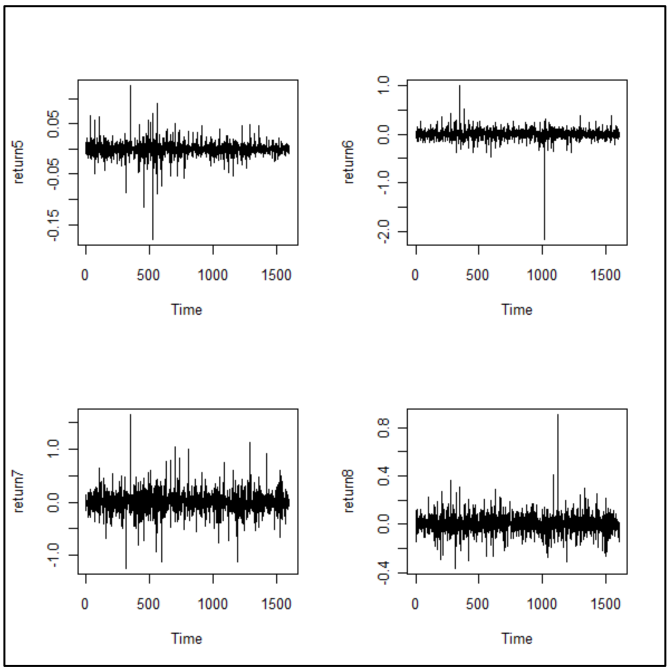


Figure 49 & 50: Returns [1] to [8]

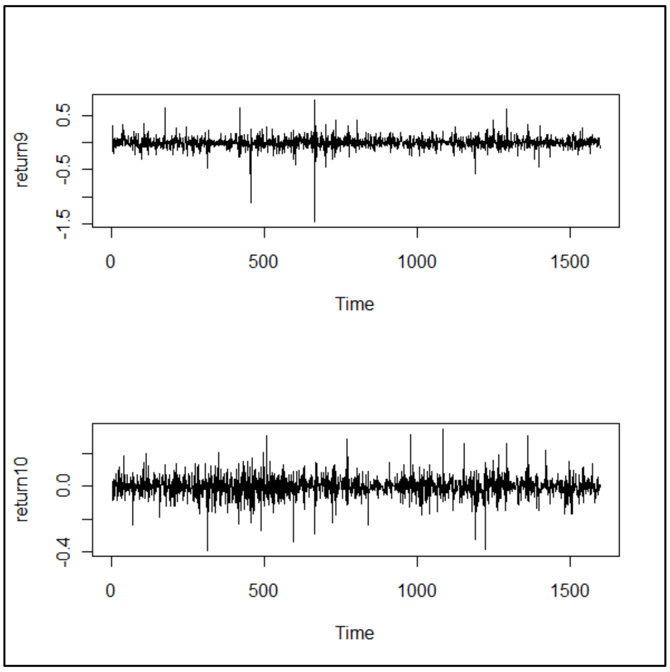
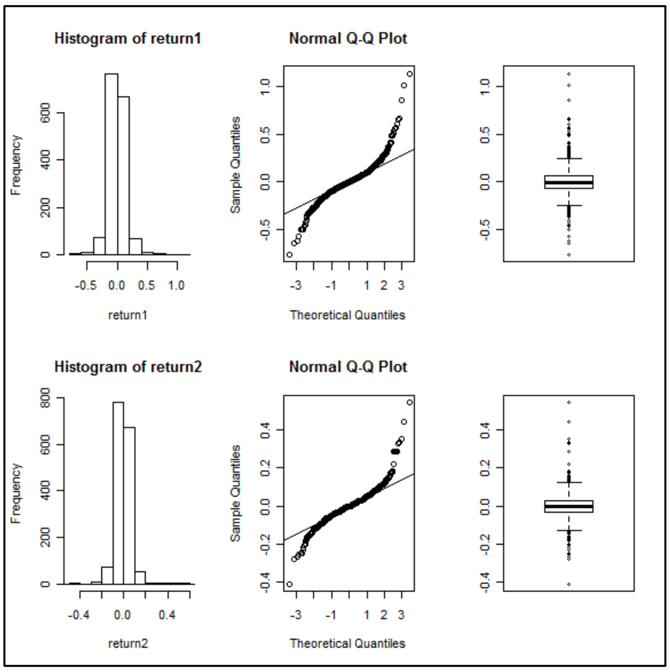


Figure 51: Returns [9] and [10]



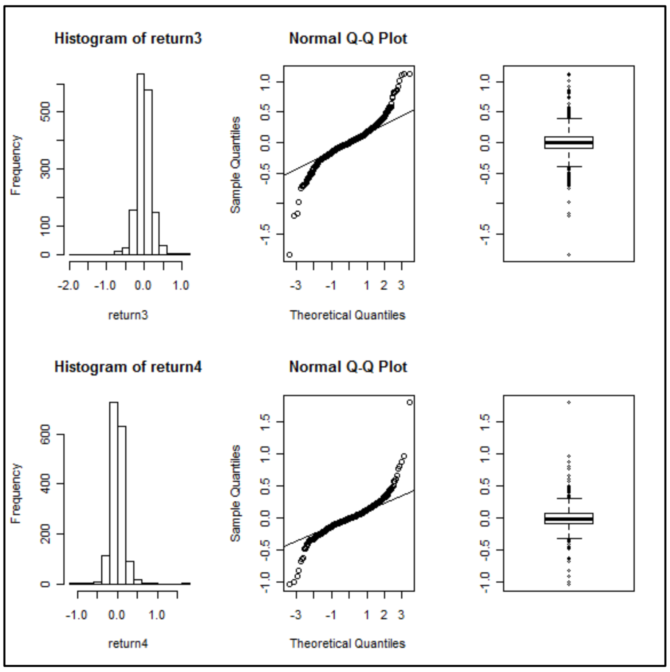
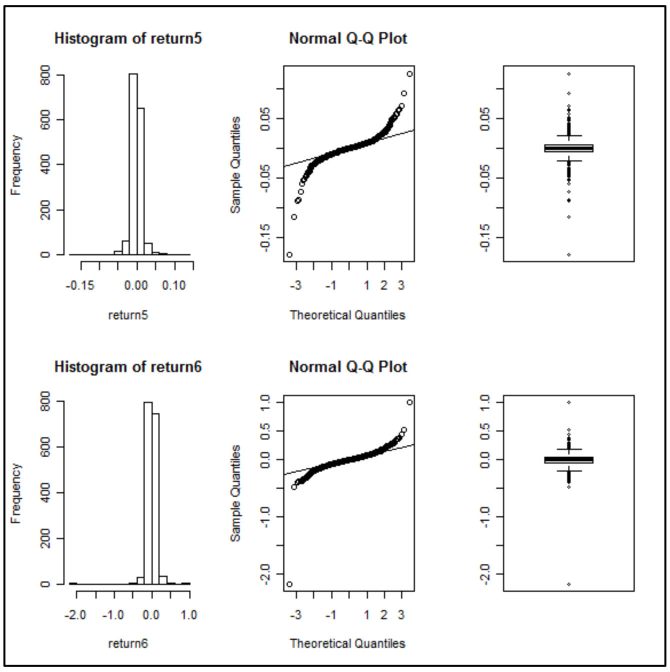


Figure 52 & 53: Statistical graphs for return [1] to [4]



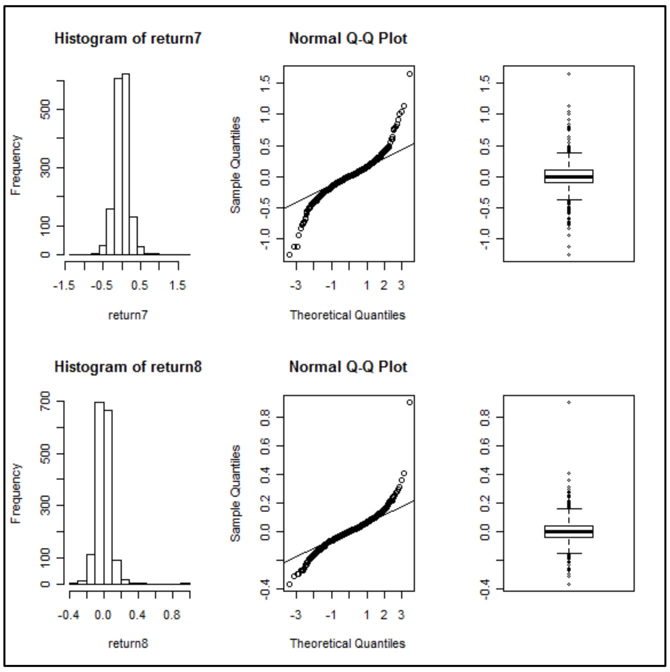


Figure 54 & 55: Statistical graphs for returns [5] to [8]

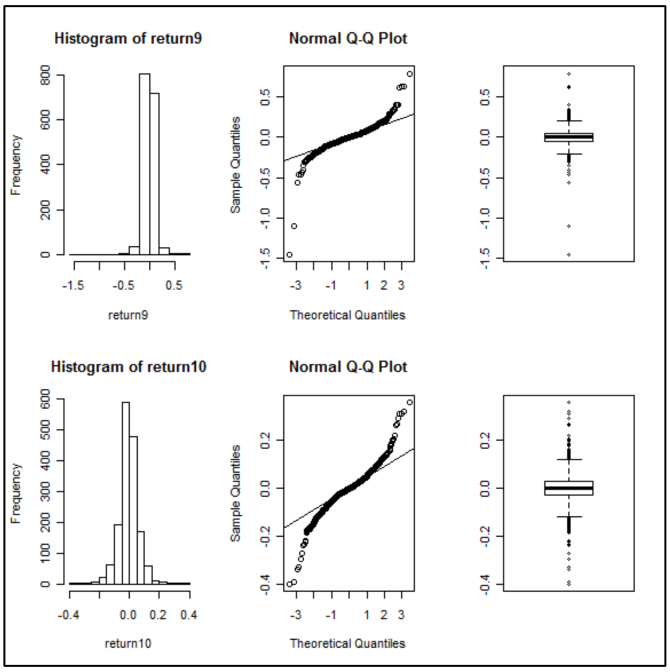


Figure 56: Statistical graphs for returns [9] & [10]

2. R codes

In this second appendix, we put the R codes of our experimentation regarding the data description, the training and the forecasting performances of the models.

a. Statistical descriptions

```
rm(list = ls(all = TRUE))
library(fNonlinear)
library(stats)
library(tseries)
op<-par(mfrow=c(3,4))
data<-read.delim2("C:/data_tesi/1.txt")
price<-ts(data[,4])
white.test(diff(price))
```

```
terasvirta.test(price)
terasvirta.test(diff(price))
white.test(price)
predict(star(price))
predict(star(diff(price)))
high<-ts(data[,2])
low<-ts(data[,3])
tick<-ts(data[,5])
volume<-ts(data[,6])
return<-diff(price)
plot(price)
hist(price)
boxplot(price)
qqnorm(price)
qqline(price)
acf(price)
pacf(price)
qqplot(volume,price)
basicStats(price)
shapiro.test(price)
decompose(price)
Box.test(price)
adf.test(price)
kpss.test(price)
acf(return)
qqplot(price,high)
qqplot(price,low)
qqplot(volume,price)
qqplot(tick,price)
adf.test(return)
kpss.test(return)
```

b. Time series models

```
rm(list = ls(all = TRUE))
library(forecast)
library(fArma)
library(fGarch)
library(tsDyn)
op<-par(mfrow=c(1,1))
data<-read.delim2("C:/data_tesi/1.txt")
Price<-ts(data[1501:1600,4])
price<-ts(data[,4])
return<-diff(price)
target_price<-price[1500:1600]
training_price<-price[1:1500]
training_return<-return[1:1499]
target_return<-return[1499:1599]
pred_price=c(1:100)
```

```
for(i in 1:100)
{
#fit_price<-HoltWinters(training_return,gamma=FALSE,beta=FALSE)
#fit_price<-star(training_return,noRegimes=2,sig=0.15)
fit_price<-armaFit(formula~arima(1,1,1),data=training_price)
#fit_price<-arfima(training_return)
#fit_price<-garchFit(~arma(1,1)+garch(2,1),data=training_return)

#s<-predict(fit_price)[1]
#pp<-predict(fit_price,n.ahead=2)
#p<-predict(fit_price)
#ppp<-predict(fit_price)
g<-predict(fit_price)[1,1]

training_price=c(training_price,target_price[i+1])
#training_return=c(training_return,target_return[i+1])

#pred_price[i]<-s
pred_price[i]<-pp$pred[1]
#pred_price[i]<-p[2]$mean[1]
#pred_price[i]<-ppp
#pred_price[i]<-g
}
accuracy(fitted.values(fit_price)[,1],return[3:1598])
#accuracy(fitted.values(fit_price)[3:1598],training_return[3:1598])
#accuracy(fitted.values(fit_price),training_return[1:1598])
#accuracy(fitted.values(fit_price),training_price[1:1599])
#accuracy(fitted.values(fit_price)[,1],training_price[3:1599])
pred_price<-ts(pred_price)
ts.plot(pred_price,target_price[2:101],gpars=list(ylab="predict(blue)_vs_target(green)",col=4:1)
#ts.plot(pred_price,target_return[2:101],gpars=
list(ylab="predict(blue)_vs_target(green)",col=4:1))
hit_p<-0
for(i in 1:100)
{if(
sign(target_price[i+1]-target_price[i])==sign(pred_price[i]-target_price[i])
#sign(target_price[i+1]-target_price[i])==sign(price[i+1500]-target_price[i])
#sign((target_return)[i+1])==sign(pred_price[i]))
{hit_p<-hit_p+1}}
hit_p<-hit_p/(length(pred_price))*100
hit_p
#accuracy(pred_price,target_return[2:101])
#accuracy(pred_price,diff(target_price))
accuracy(pred_price,target_price[2:101])

#price[1501:1600]<-price[1500:1599]+pred_price
#accuracy(price[1501:1600],Price)
```

c. NN1

```
rm(list = ls(all = TRUE))
library(forecast)
library(monmlp)
data<-read.delim2("C:/data_tesi/1.txt")
price<-data[,4]
return<-diff(price)
op<-par(mfrow=c(1,1))
r5<-return[5:1498]
r4<-return[4:1497]
r3<-return[3:1496]
r2<-return[2:1495]
r1<-return[1:1494]
input_NN_training<-cbind(r1,r2,r3,r4,r5)
input_NN_test<cbind(return[1495:1594],return[1496:1595],
return[1497:1596],return[1498:1597],return[1499:1598])
target_training<-data.matrix(return[6:1499])
target_test<-data.matrix(return[1499:1599])
MLP_price<-monmlp.fit(input_NN_training,target_training,hidden1=3)
accuracy(attr(MLP_price,"y.pred"),target_training)
pred_price<-ts(monmlp.predict(input_NN_test,MLP_price))
ts.plot(ts(pred_price),ts(target_test[2:101]),
gpars=list(ylab="predict(blue)_vs_target(green)",col=4:1))
hit<-0
#Price<-ts(data[1500:1600,4])
for(i in 1:100)
{if(
#sign(Price[i+1]-Price[i])==sign(price[i+1500]-Price[i])
sign(return[i+1499])==sign(pred_price[i]))
{hit<-hit+1}}
hit<-hit/(length(pred_price))*100
hit
accuracy(pred_price,target_test[2:101])
Price<-ts(data[1501:1600,4])
price[1501:1600]<-price[1500:1599]+pred_price
accuracy(price[1501:1600],Price)
```

d. NN2

```
rm(list = ls(all = TRUE))
library(forecast)
library(monmlp)
data<-read.delim2("C:/data_tesi/1.txt")
op<-par(mfrow=c(1,1))
price<-data[,4]
return<-diff(price)
high<-data[3:1499,2]
```

```
low<-data[3:1499,3]
open<-data[3:1499,4]
volume<-data[3:1499,6]
high_test<-data[1500:1599,2]
low_test<-data[1500:1599,3]
open_test<-data[1500:1599,4]
volume_test<-data[1500:1599,6]
r2_test<-return[1499:1598]
r1_test<-return[1498:1597]
input_NN_training<-cbind(high,low,volume,return[2:1498],open)
input_NN_test<-cbind(high_test,low_test,volume_test,r2_test,open_test)
target_training<-data.matrix(return[3:1499])
MLP_price<-monmlp.fit(input_NN_training,target_training,hidden1=3,n.trials=2)
accuracy(attr(MLP_price,"y.pred"),target_training)
target_test<-return[1499:1599]
predict_price<-ts(monmlp.predict(input_NN_test,MLP_price))
ts.plot(ts(predict_price),ts(target_test[2:101]),gpars=
list(ylab="predict(blue)_vs_target(green)",col=4:1))
hit<-0
Price<-ts(data[1500:1600,4])
for(i in 1:100)
{if(
#sign(Price[i+1]-Price[i])==sign(price[i+1500]-Price[i])
sign(return[i+1499])==sign(predict_price[i]))
{hit<-hit+1}
}
hit<-hit/(length(predict_price))*100
hit
accuracy(predict_price,target_test[2:101])
Price<-ts(data[1501:1600,4])
price[1501:1600]<-price[1500:1599]+predict_price
accuracy(price[1501:1600],Price)
```

BIBLIOGRAPHY

The title is written first and in italic, then we can find the name of the authors and finally the source of the paper like a specific journal.

[1] *25 years of time series forecasting*. De Gooijer et al., International Journal of Forecasting 22 (2006) 443 – 473.

[2] *A comparison of the accuracy of short term foreign exchange forecasting methods*. Meade, International Journal of Forecasting 18 (2002) 67 – 83.

[3] *A hybrid ARIMA and support vector machines model in stock price forecasting*. Pai et al., Omega 33 (2005) 497 – 505.

[4] *An empirical investigation of the usefulness of ARFIMA models for predicting macroeconomic and financial time series*. Bhardwaj et al., Journal of Econometrics 131 (2006) 539 – 578.

[5] *An Introduction to High-Frequency Finance*. Dacorogna et al., International Review of Economics and Finance 12 (2003) 525 – 529.

[6] *An introduction to statistical finance*. Bouchaud, Physica A 313 (2002) 238 – 251.

[7] *Designing a neural network for forecasting financial and economic time series*. Kaastra et al., Neurocomputing 10 (1996) 215 – 236.

[8] *Distinguishing between stochastic and deterministic behaviour in high frequency foreign exchange rate returns: Can non-linear dynamics help forecasting?* Cecen et al., International Journal of Forecasting 12 (1996) 465 – 473.

[9] *How Profitable Are High-Frequency Strategies?* Irene Aldridge, the Huffington Post, July 26, 2010.

[10] *Dynamical structures of high-frequency financial data*. Kim et al., Physica A 376 (2007) 525 – 531.

[11] *Effective multi-fractal features of high-frequency price fluctuations time series and l-variability diagrams*. De Souza et al., Chaos, Solitons and Fractals 42 (2009) 2512 – 2521.

[12] *Efficient market hypothesis and forecasting*. Timmermann et al., International Journal of Forecasting 20 (2004) 15 – 27.

[13] *Financial econometric analysis at ultra-high frequency: Data handling concerns*. Brownlees et al., Computational Statistics & Data Analysis 51 (2006) 2232 – 2245.

[14] *Financial time series forecasting using independent component analysis and support vector regression*. Lu et al., Decision Support Systems 47 (2009) 115 – 125.

[15] *Financial time series forecasting using support vector machines*. Kim, Neurocomputing 55 (2003) 307 – 319.

[16] *Finite Mixture of ARMA-GARCH Model for Stock Price Prediction*. Tang et al., Proc. of 3rd International Workshop on Computational Intelligence in Economics and Finance (CIEF'2003), North Carolina, USA, September 26 - 30, 2003, pp. 1112 - 1119.

[17] *Forecasting economic and financial time-series with non-linear models*. Clements et al., International Journal of Forecasting 20 (2004) 169 – 183.

[18] *High frequency data in financial markets: Issues and applications*. Goodhart et al., Journal of Empirical Finance 4 (1997) 73 – 114.

[19] *Is the predictability of emerging and developed stock markets really exploitable?* Moreno et al., European Journal of Operational Research 182 (2007) 436 – 454.

[20] *Naive, ARIMA, nonparametric, transfer function and VAR models: A comparison of forecasting performance*. Thomakos et al., International Journal of Forecasting 20 (2004) 53 – 67.

[21] *Neural network modelling for stock movement prediction: A state of the art*. Coupelon.

[22] *Non-linear forecasting of stock returns: Does volume help?* McMillan, International Journal of Forecasting 23 (2007) 115 – 126.

[23] *Nonlinearities, cyclical behaviour and predictability in stock markets: international evidence*. Sarantis, International Journal of Forecasting 17 (2001) 459 – 482.

[24] *Nonlinearity and intraday efficiency tests on energy futures markets*. Wang et al., Energy Economics 32 (2010) 496 – 503.

[25] *STAR and ANN models: forecasting performance on the Spanish « Ibex-35 » stock index*. Pérez-Rodriguez et al., Journal of Empirical Finance 12 (2005) 490 – 509.

[26] *Statistical properties, dynamic conditional correlation and scaling analysis: Evidence from Dow Jones and Nasdaq high-frequency data*. C. Chiang et al., Physica A 388 (2009) 1555 – 1570.

[27] *Surveying stock market forecasting techniques – Part II: Soft computing methods.* Atsalakis et al., Expert Systems with Applications 36 (2009) 5932 – 5941.

[28] *Business Intelligence.* Carlo Vercellis, McGraw-Will edition, 2005.

[29] *Modelling Financial Time Series*. Stephen J. Taylor, A Wiley Finance Edition (page 1 – 61).

[30] *Neural Network, Time series forecasting of financial markets*. Michael Azoff, A Wiley Finance Edition (page 1 – 90).

[31] *Comparing linear and nonlinear forecasts for stock returns*. Kanas et al., International Review of Economics and Finance 10 (2001) 383 – 398.

[32] *Neural networks for technical analysis: a study on KLCI.* YAO et al., International Journal of Theoretical and Applied Finance Vol. 2, No. 2 (1999) 221 – 241.

[33] CRAN web site for R packages and tools description.

[34] Wikipedia.

[35] *Testing for Long Memory in ISE Using ARFIMA-FIGARCH Model and Structural Break Test*. Korkmaz et al., International Research Journal of Finance and Economics; ISSN 1450-2887 Issue 26 (2009).

[36] *Prévision ARFIMA des taux de change: les modélisateurs doivent-ils encore exhorter à la naïveté des prévisions?* LARDIC et al., ANNALES D'ÉCONOMIE ET DE STATISTIQUE, N° 54 – 1999.

[37] *Statistical analysis of high frequency data from the Athens stock exchange*. Mills, Physica A 293 (2001) 566 – 572.

[38] *Market microstructure: a survey*. Madhavan, Journal of Financial Markets 3 (2000) 205 – 258.

[39] *Behavioural finance and asset prices: Where do we stand?* Stracca, Journal of Economic Psychology 25 (2004) 373 – 405.

[40] *A fusion model of HMM, ANN and GA for stock market forecasting*. Hassan et al., Expert Systems with Applications 33 (2007) 171 – 180.

[41] *Another look at the forecast performance of ARFIMA models*. Ellis et al., International Review of Financial Analysis 13 (2004) 63 – 81.

[42] *Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index*. Chen et al., Computers & Operations Research 30 (2003) 901 – 923.

[43] *Forecasting High-frequency Financial Data with the ARFIMA–ARCH model*. Hauser et al., Journal of Forecasting J. Forecast. 20, 501 – 518 (2001).

[44] *Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait*. Mostafa, Expert Systems with Applications 37 (2010) 6302 – 6309.

[45] *Forecasting stock market movement direction with support vector machine.* Huang et al., Computers & Operations Research 32 (2005) 2513 – 2522.

[46] *Forecasting stock market short-term trends using a neuro-fuzzy based methodology*. Atsalakis, Expert Systems with Applications 36 (2009) 10696 – 10707.

[47] *Intraday patterns and local predictability of high-frequency financial time series*. Molgedey et al., Physica A 287 (2000) 420 – 428.

[48] *Non-linear forecasting in high-frequency financial time series*. Strozzi et al, Physica A 353 (2005) 463 – 479.

[49] *Non-linear logit models for high-frequency data analysis*. Sazuka, Physica A 355 (2005) 183 – 189.

[50] *Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods*. De Faria, Expert Systems with Applications 36 (2009) 12506 – 12509.

[51] *Seize the(intra)day: Features selection and rules extraction for tradings on high-frequency data.* Resta, Neurocomputing 72 (2009) 3413–3427.

[52] *The use of data mining and neural networks for forecasting stock market returns*. Enke et al., Expert Systems with Applications 29 (2005) 927 – 940.

[53] *Empirical properties for asset returns: stylised facts and statistical issues*. Cont, Quantitative finance volume 1 (2001) 223 – 236.