ALGORITHMIC TRADING TECHNOLOGY AND STRATEGY RESEARCH ON FINANCIAL MARKETS

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Degree Programme in Business Information Technology

Visamäki, spring 2017

Victor A. Barca

Degree Programme in Business Information Technology
Visamäki Campus

Author	Victor A. Barca	**Year** 2017
Subject	Algorithmic Trading Technology and Strategy Research on Financial Markets	
Supervisor(s)	Lasse Seppänen	

ABSTRACT

Due to the financial digitalization, the human intervention in many operations both functional and those aimed to make profit in the market are likely to be substituted by algorithms and automated processes.

This thesis lays the foundations of algorithmic trading developing a framework to automate the whole process of trading financial assets. Also, it researches trading strategies and the real potential to make profit.

The framework consists of separated functional modules to acquire market data, to provide performance ratios and plot visualizations in the research and develop of trading strategies and to integrate these strategies with the stock exchange, allowing the automated communication in between.

The framework developed, proves the power of the present and future relationship between finance and digital technology. However, exploiting market inefficiencies through trading liquid assets, is a daunting challenge that requires an arms race of ideas and strategies but, for sure, it will continue pushing the boundaries of the technology.

Keywords Finance Technology, FinTech, Algorithmic Trading, Market Efficiency.

Pages 44 pages including appendices 5 pages

CONTENTS

1 INTRODUCTION

Since the 1970s, the financial institutions have been increasing the usage of digital technology for operations sensitive to be automated. (Wyss, 2000, p. 4)

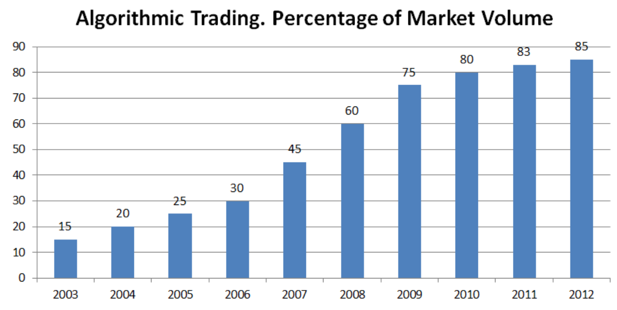


Figure 1 *Algorithmic trading. Market volume (Glantz & Kissell, p. 258)*

Due to the competition having the latest technology developing efficient and secure systems, many operations that were done manually in the past cannot be done in this way anymore due to the complexity and speed required to accomplish it (FT, 2016)

The information generated by the economy (news, company earnings, etc.) is quickly included to the market modifying the prices. Predicting future movements in prices could be not possible, because it means, predict this information.

This thesis will be focused in the type of operations that have the goal of making a profit in the market, also known as Quantitative trading. The main objective is the study and development of technology that make possible the algorithmic trading and design algorithmic strategies capable of operating real markets without human interventions.

The knowledge acquired in this thesis and the framework to implement will be aimed to personal investors or investing companies that want to gain insight in algorithmic trading as well as the research process in order to develop automated trading strategies

The main questions by which the objectives will be achieved are:

- How can the market data be obtained from an external resource to be automatically processed?
- How can a given trading strategy be transformed into an algorithm?
- How can orders, given by an algorithm, be transmitted to a stock exchange?
- When is an algorithm strategy really profitable?
- Is it possible to make profit with algorithmic trading?

2 FINANCE AS TECHNOLOGY

The evolution of finance could be described like any other process of invention of new technology. Someone has a necessity, then, makes a prototype that solve a problem and if it works, it rapidly spreads around the world, making others improve and modify the flaws of the initial design. Eventually, the instruments designed reach such a high level of complexity that could be compared with any other piece of engineering in any other field.

The finance technology has had to solve many problems related to key necessities of humankind since the beginning of time. This made possible the collaboration among people around the world to accomplish challenging objectives or reducing uncertainty in the future when is dealing with resources and property.

Due to the origin of human communication in prehistoric times, the possibility of exchanging products and services between humans started. This initial form of trade was through barter those products without any other element involve (Watson, 2005). With the passing of time the trades evolve to use commodity money, an object with valuable characteristic for a specific community in a specific time and place. The gold was the ultimate commodity money which finally evolves into the modern money that it is known nowadays in its form of paper and coins. The next step in the evolution of money could be the cryptocurrencies like Bitcoin.

To trade it was necessary to have a good, service or money in exchange and have it right in that moment. However, sometimes the cost of products is too high to be able to afford it. The loans were evolved together with the history of trade and money to acquire those products or even to embark in big ventures. The sense of modern banking was born in the 14th century in Italy where the money was available as a good (Macesich, 2000). The banks allow to exchange money in return of an interest, an amount of money that has to be paid in the future by the borrower. The source of the borrowed money could be from many agents, called investors, interested in a part of the future profit of the deal. This way of project finance brings the advantage of reducing the risk among many investors but open the door to systemic disasters if that risk is not correctly measurable.

If the people is pursuing a common objective and they need money for it, they could start a company. A company is a way to identify a purpose and attract money to build the necessary elements to achieve that purpose. Although, there are evidences that since the Roman Empire an investor could participate in a company and trade these participations called shares (Malmendier, 2008) the first most famous company to have tradeable shares was the Dutch East India Company founded in 1602 to transport goods in dangerous waters (de Vries & van der Woude, 1997). A share is the right of receiving a portion of the company's profit, the invested money will be part of the company’s assets called capital. Hence, a company could ask for capital issuing shares and a potential investor receive dividends for the future company’s profit.

The investors can also sell or buy shares from another investor. The places to do these trades are called stock markets. The trade of shares open the gate to as many interpretations as investors are about the share’s price, developing the techniques to evaluate a company, forecast the future returns and measure the risk involve in an investment.

The risk is something inherent to any human activity and to protect a property, trade or venture from a natural or deliberate disaster, insurance was created. Insurance could be a law or contract that allow compensation if something happened, reducing the possible future profit or the final cost of a product but eliminating the risk of losing everything. Insurance has evolved as technology since ancient times but the modern form of insurance born in London due the “Great Fire of London” in the 17th century and the develop of probability (Dickson, 1960).

Specifically, to protect trades against the movement of prices, derivatives were formed. The derivatives are contracts over an underlying asset where two parties specify the conditions (dates, price, etc.) under which the trade have to be made. The futures are derivatives where one part promise today for a fix payment to deliver the underlying in the future. The options are derivatives where the buyer of the option has the right to buy or to sell the underlying for fixing a payment, but the seller has the obligation (Koehler, 2011).

The latest technology invented in the finance world was the decentralized cryptocurrencies. In 2009, the cryptocurrency Bitcoin appear as digital currency and payment system where the ledger or list of transactions ever made, called Blockchain, is public and maintained by different nodes around the world hence, cannot be controlled by any government. (Brito & Castillo, 2013)

3 PRICES AND EFFICIENCY

The price of an asset is the principal piece of information of any market, representing the monetary value that two agents agreed to trade in the past. This is an important point to operate the markets, this information, the price, is always historical, a certain monetary value of a trade made in the past. In a competitive market, there are two sides, those who are willing to sell and those who are willing to buy and both have in mind a price. When these prices, bid and ask respectively, coincide, the transaction takes place and this price becomes part of the past. This price is usually shown in charts and financial news. The difference between bid and ask is called spread.

To make a profitable approach to the market it is important to understand what the price means and how is created, not technically but in the mind of the agents. For the economist F. Hayek the price system is mechanism of communicate information:

"We must look at the price system as such a mechanism for communicating information if we want to understand its real function... …It is more than a metaphor to describe the price system as a kind of machinery for registering change, or a system of telecommunications which enables individual producers to watch merely the movement of a few pointers, as an engineer might watch the hands of a few dials, in order to adjust their activities to changes of which they may never know more than is reflected in the price movement."

(Hayek, 1945)

Therefore, the variations in price of an asset comes from the new information aggregated by the agents that are participating in that market. The market efficiency could be described as how fast and all the information generated is included in the price. Hence, in a competitive market where many investors are trying to include that information, and making profit of it, rapidly the possibility of making profit for other is reduced to 0, reaching again, the previous intrinsic value where all the information past and present is included.

3.1 Efficient Market Hypothesis

The Efficient Market Hypothesis (EHM) developed by the economist E. Fama (1970) claims that all the known information about an asset is already fully aggregated to its price. This hypothesis has three forms and a theoretical implication for each one of them:

Table 1 *Efficient Market Hypothesis forms (Fama, 1970)*

Version	The prices include	Impossibility making profit from
Weak-form	All the past information	Technical analysis
Semi-strong	All the past and immediately present public information.	Fundamental analysis
Strong	All the past, present and information only known insider to the company, not public.	Privilege information

When it is said that an investor could not make profit it means that they cannot have “excess return”, “risk-adjusted returns” or “beat the market”. In other words, it cannot make more returns than the average market or benchmark of reference thus, any strategy more than buy and hold an index will be worthless. For example, if a trading strategy make 7% a year but the whole market or reference benchmark make 10% the strategy is no really profitable assuming the same level of risk.

The implication of the EHM in this thesis is very serious. As the new information is randomly generated, no one knows whether will be good or bad, the prices could be describing a “random-walk”, making impossible predict return (Fama, 1965).

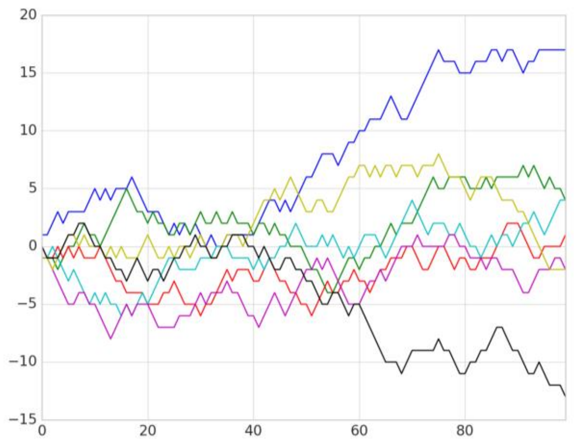


Figure 2 *Random walk [Appendix 1]*

The EHM describes a rational market where it only reacts to new information or changes of the intrinsic value of an asset, its present value (discount rates). However, many investors and researchers have disputed this claim saying that there are individuals “beating the market” and that the market most of the time, it is irrational (Buffet, 1984). In 1981, the economist Robert Shiller published an article showing that the volatility of the returns of the U.S. market since 1920 has been greater than any “rational” explanation expected by the dividends that those companies gave in the posterior period (YaleNews, 2011).

3.2 Behavioural finance

Behavioural finance or behavioural economic has emerged from those researches that found inefficiencies in the markets in opposite to the EMH says. When investors make an approach to markets they bring with them cognitive limitations making systematic errors at the time of evaluating assets rationally. These irrational assumptions could lead to overvalue or undervalue assets beyond from its "intrinsic value" o "fair price" opening the gate to exploit that inefficiencies. (Lin, 2012). Those limitations are:

Table 2 *Limitations of human cognition (Lin, 2012)*

Group	Type	Tendency
Cognitive Biases	Overconfidence	The future will be much better than it finally is.
	Status quo bias	The same choices made it in the past are good choices now.
	Loss aversion	To feel worse a loss than a gain of the same amount
	Endowment effect	To assign more value to something that is already owned than when it is not
	Confirmation bias	To search information that confirms of our personal thoughts and preferences and reject opposite opinions
Heuristics	Anchoring	To include unrelated prior information to the analysis process.
	Availability	To see more likely to happen events easier to imagine than other with the same probability but difficult of imagine.
	Representativeness	To analyse and contrast something from another similar thing that it is already known.
	Herd behaviour	To follow trends or to think in a particular way because others are doing the same
Framing Effect	Context	To modify the perception of something depending on how is the information it is presented.

The purpose of any **trading strategy** is to exploit those inefficiencies but having in mind that any already known and profitable strategy could not work anymore because the use for other investors.

4 ALGORITHMIC TRADING

Since 1990, the digitalization of the finances brings the ability to trades shares, before done by physically be in the stock exchanges, now trough internet with online brokers. The broker has to provide a way to allow the communications computer-to-broker not just only human-to-broker. Any broker nowadays provides computer applications to manually set orders that will be transmitted to the market but also some of them, provide an Application Program Interface (API). An API is an interface with computer procedures to communicate the broker with and external program. This technology allows the creation of algorithms that automatically operate the market. (Lin, 2014).

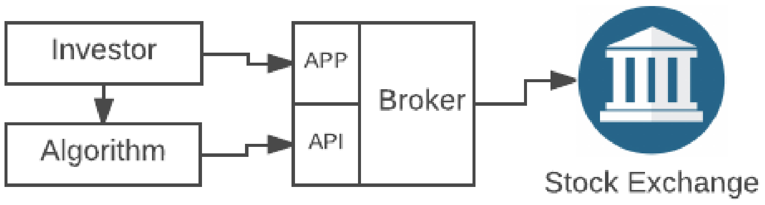


***Figure 3** Algorithmic Trading Relations*

With algorithmic trading, two types of operations could be distinguished depending on the financial agent and the purpose. On the one hand, operations aimed at minimizing cost, market impact or risk, for example sell a large number of shares in different quantities and periods to reduce the impact in the price of the company. On the other hand, operations aimed to make profit in the market like market making, arbitrage or speculation. These operations are commonly known as Quantitative trading.

Quantitative trading is the trading of shares, derivatives and other financial instruments trough computer algorithms. Any trading idea suitable of be designed as a computer algorithms enter in the field of quantitative trading, in particular, those with a clear mathematical definition and not much other like technical analysis where the figures and patterns in prices are difficult to code. (Chan, 2008).

Those computer algorithms use historical data like prices, trading volume, financial statements, ratios, etc. to test trade conditions and decide if it is a good strategy for real data based in the historical performance.

4.1 **Automated Trading Strategy**

An automated trading strategy comprises the steps or conditions to buy or sell assets that will be translate into an algorithm. The result of a strategy will be the returns over time of those trades and the performance measurements.

Mainly, there are two kinds of strategies to perform in quantitative trading, mean-reversion and momentum. If the price of an asset is relative low or high respect a reference, for example the average historical price of a share, and it is expected the come back to that reference, the strategies exploiting this situation are called mean-reversion strategies. For example, it is possible to make a strategy with two stocks highly correlated in time. If the prices are moving in opposite direction (making the difference between them greater) buying one and selling the other with the same amount of money expecting a reverse in that movement. The same with assets negatively correlated.

However, if the price of an asset moves away from the reference it could be exploited with momentum strategies that follow a trend in a certain direction. For example, strategies that try to identify trend changes with technical indicators or strategies that use fundamental data to reach certain price objective.

4.2 **Performance ratios**

The ratios of performances are present in the whole cycle of testing to ensure the performance will be good at time to performs real trades. They allow to decide if a strategy can go further or not in the research process.

Usually in the media or investment fund reports it is shown the returns to tell how an investment has been performed in a period of time but this measure says little of the characteristics and risk the investment had.

Things like long periods of negative returns or high returns following of big drops could we dangerous in terms of psychology and capital leverage.

4.2.1 **Sharpe Ratio**

The latest definition of the Sharpe Ratio measures the excess return of the portfolio in terms of excess portfolio's volatility. (Sharpe, 1994)

$$SR = \frac{R_p - R_b}{\hat{\sigma}_{p-b}}$$

Where:

R_p = portfolio or strategy return.
R_b = benchmark return.
$\hat{\sigma}_{p-b}$ = standard deviation of the difference in returns between the portfolio and its benchmark, excess volatility.

This ratio measures just the alpha component of the total return**.** As it was said before, the "excess return" attributed to the strategy itself, and not to the market. This total return could be divided in three components, risk free return, beta return, and alpha return:

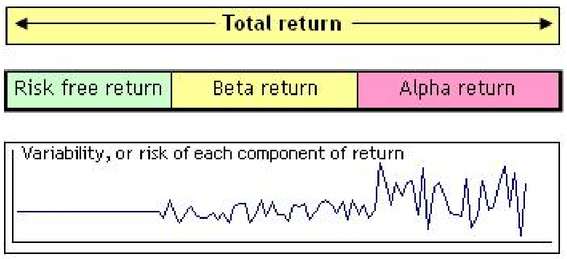


Figure 4 *Returns and volatility division (Pareek, 2008)*

Each component adds a level of returns and volatility that the strategy's performance must beat.

Table 3 *Risk components*

Component	Description	Return added	Risk added
Risk-Free	Investment risk-free like treasury bonds.	Bond	0%
Beta	Market or reference benchmark	R_b	$\hat{\sigma}_b$
Alpha	Active management	$R_p - R_b$	$\hat{\sigma}_{p-b}$

Important to note that the Sharpe Ratio does not distinguish between positive and negative returns at time to measure the volatility. In fact, higher positive returns will increase the volatility and decrease the ratio for that strategy.

4.2.2 **Sortino Ratio**

The Sortino Ratio is a modification of the Sharpe Ratio where only the downside volatility penalizes the ratio (Red Rock). This gives a better measure contrasting strategies. For example, if it has one strategy with high returns and low downsides and another with a decent return and low volatility, the Sharpe ratio will assign better measure to the lowest volatility strategy when the other one is better in all aspects.

$$S = \frac{R - T}{TDD}$$

Where:
R = average return
T = desire minimum return
TDD = target downside deviation, volatility of the returns below the desire minimum return.

4.2.3 **Drawdown measures**

- Max drawdown is the maximum fall we encountered in the whole period. It could be in a long period or short period of time, like a flash crash.

- Max drawdown period is maximum period in which we starting losing returns we had acquired.

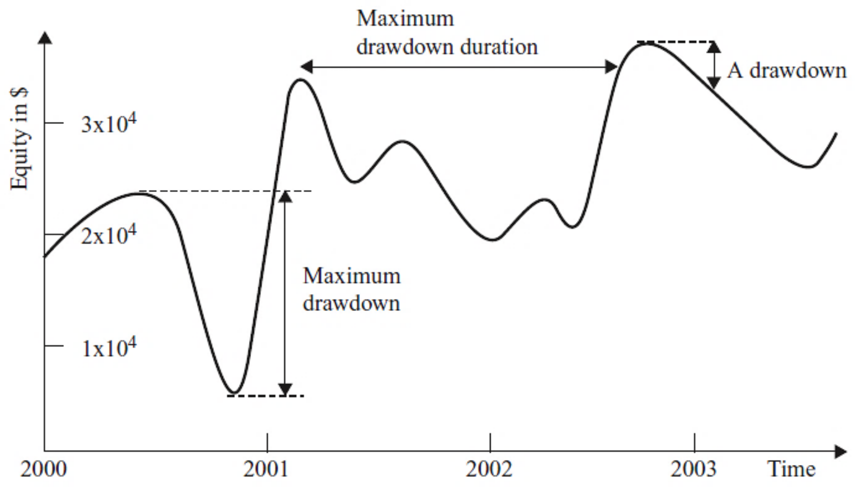


Figure 5 *Drawdowns (Chan, 2008)*

4.2.4 **Other measures**

Period return is the relation between the initial capital that was invested and the profit made it, getting the percentage of return of the trading strategy. Profit per trade is the average return per transaction made it. Profit per $ is the average return per monetary unit used.

4.3 **Technical indicators**

Any market provides data about trades already made, prices and volume in different periods of time (weekly, daily, per second, etc.). A technical indicator is a mathematical construction that takes as input this basic information to help a trading strategy to forecast future prices.

Those indicators could be divided into two types, trend indicators like moving averages moves with the price and can have any value whereas the oscillators move between a range, generally 0 to 100, like the RSI and MACD.

4.3.1 Simple and Exponential Moving Average (SMA, EMA)

The simple moving average (SMA) smooth the price movement taking the average of a certain window of periods backwards. Depending on these windows, it is said that the moving average is the long-term or the shot-term, being for the price more or less strong break over or down. For this reason, some stocks seem to “fit” some moving averages useful to identify future trends.

The exponential moving average (EMA), unlike the SMA, applies more weight to latest prices hence, the “reaction” of the indicator is faster than the SMA.

$$Multiplier = \frac{2}{window + 1}$$

$$EMA^{t} = (price - EMA^{t-1}) * Multiplier + EMA^{t-1}$$

(StockChart.com)

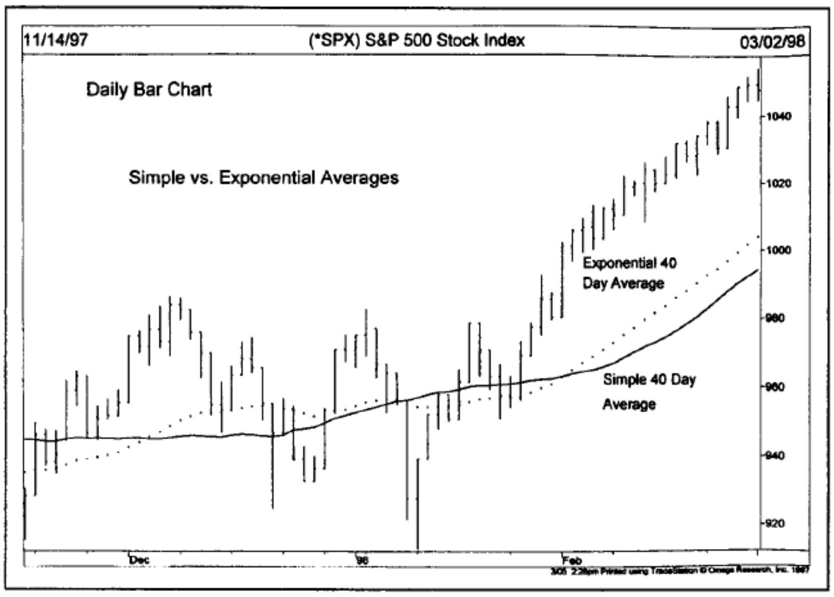


Figure 6 *Moving averages (Murphy, 1999)*

4.3.2 Relative Strength Index (RSI)

The relative strength index (RSI) is a popular oscillator that compares the magnitude of gains and losses of an asset's price in a certain period of time and has a scale of 0 to 100. if the indicator has a high value, it is said that the asset is overbought whereas the value is low it is said that the asset is oversold.

$$RSI = 100 - \frac{100}{1 + RS}$$

$$RS = Average\ Gain\ /\ Average\ Loss$$

The most common configuration for this indicator is a window of 14 periods an upper limit of 70 and a lower limit 30. (Murphy, 1999, pp. 239-246).

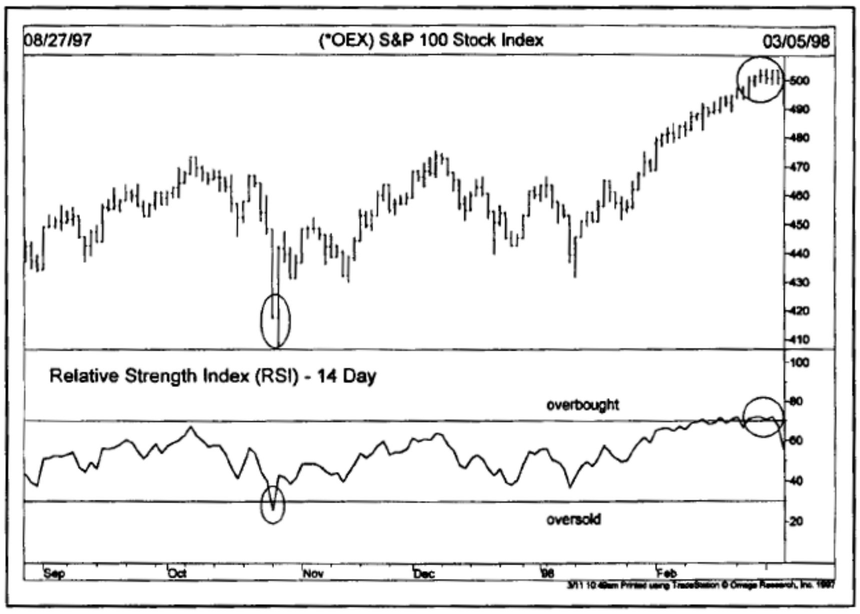


Figure 7 *RSI (Murphy, 1999)*

A trading strategy can use this indicator as signal of support or resistance in prices and it is particularly useful with stocks in lateral move rather than a trend.

4.3.3 Moving Average Convergence Divergence (MACD)

The Moving Average Convergence Divergence (MACD) is an oscillator that consists of two lines. The faster line o "MACD Line" is the difference between two EMA of 12 and 26 periods. The slower o "Signal line" is an EMA of 9 periods.

The signals are given when the two lines cross. A buy signal when MACD line is crossed from above by the Signal line and a sell signal when the MACD line is crossed from below. The interpretation of the indicator is that prices are "overbought" when are far above from the zero line and "over sold" when the lines are far below from the zero line.

The numbers 12, 26, and 9 the common used periods for this indicator. (Murphy, 1999, pp. 252-254).

$$MACD\ line = EMA12(prices) - EMA26(prices)$$

$$Signal\ line = EMA9(MACD)$$

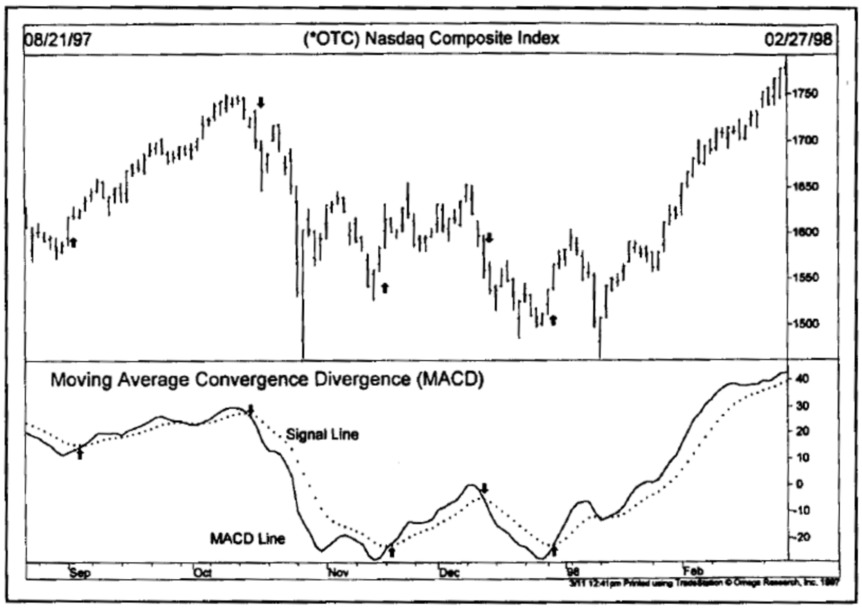


Figure 8 *MACD (Murphy, 1999)*

Both the RSI and MACD can be used as a confirmation of movement. When the prices go in one direction and the oscillator to another it is said that there is a "divergence". The divergence is a disagreement and it can indicate that something in changing.

5 STRATEGY RESEARCH METHODOLOGY

The knowledge acquired until now gives the fundamental theory about the surrounding elements around algorithmic trading: market efficiency, measures of the market as indicators, common types of strategies used to exploit inefficiencies and performance ratios to describe the results of those strategies. Now it is time to develop a strategy.

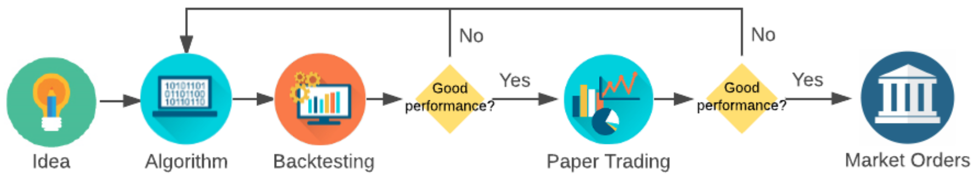


Figure 9 *Cycle of a Trading Strategy Research*

This research cycle is designed to ensure a profitable strategy testing an idea through different levels of data until there is enough trust to operate the market. First, testing the strategies with historical data, easy and fast to access to quickly discard bad ideas. If the strategy pass that stage, is tested with real time data but without perform any real order, this stage takes more time because the information must be generated by the market in the periods of time the strategy was designed. Finally, if the strategy is consistency with the result given in early stages it is ready to make real operations (Tulchinsky, 2015, pp. 27-30)

5.1 Idea

At the time to design a profitable trading strategy it must be taken into account aspects both from computational and trading point of view. An idea could be possible to perform with human intervention such as interpreting financial statements of a company, assess a brand and its appreciation for the costumers or even looking for patterns in price charts looking for future returns. However, those ideas are difficult to encode in form of computer algorithms.

The suitable ideas for a computer algorithms are those with a clear mathematical, numerical or statistical form. For example, when the price goes up more than 10% in a week the following week goes down 3% so the strategy will be looking for this initial condition selling at that point, hoping to buy when the price reach that 3% closing the position. If a condition is met, what is normal to happen in the following weeks or months? Is this happening in a particular stock, sector group or the whole market? Those kinds of questions are the focus and first step of a trading strategy research.

The data used will vary depending on the characteristic of the idea. A strategy could use intra-day, weekly, monthly prices, etc. The conditions could be based on prices, volume, news, fundamental ratios, etc. Any source of data able to be processes for a computer could be suitable.

5.2 Testing environments

The initial idea must be tested to see whether it is true or not. It is possible to do statistical research with this historical data to look for statistical behaviors or simulate the strategy to see how it would have performed in the past, this is called backtesting (Chan, 2008, p. 31). The backtesting is an environment where the algorithm can work with historical data simulating the market, a virtual market. It could be added parameters, hidden costs, etc. to get an environment as similar as possible with the real market.

The backtesting is the first filter the strategy has to pass to continue the research cycle. Depending on the risk it be wanted to take, If the result's performance is not good, for example, the strategy has big drawdowns, the idea must be redesigned or change some parameters and test again. If the strategy is good can get the next level of testing, paper trading.

Paper trading is testing the trading strategy not with historical data but the data generated in real time for the market. Real orders are transmitted to the broker to a special account that simulates the process of making trades with the stock exchange, including virtual money. The advantage of this stage is the feedback given by the broker by including real transaction costs, times of operations, performance measures, etc. It the closest environment to the market.

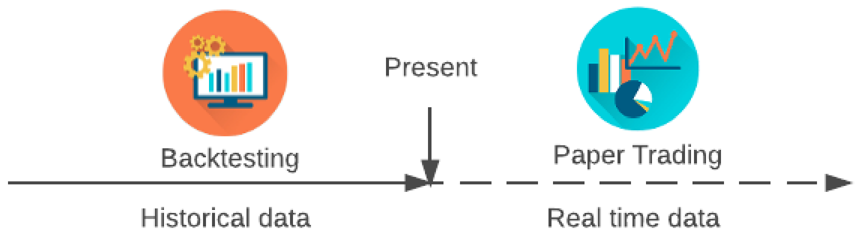


Figure 10 *Testing environments differences*

5.3 Pitfalls to avoid

In the strategy research process, there are numerous ways to fall in pitfalls, leading in well performed strategies with historical data but useless with real-time prices.

Table 4 *Issues when testing a strategie (Chan, 2008, p. 67)*

Scope	Check
Data	Split and dividends must be correctly adjusted.
	Missing data and abnormally differences between periods.
	Survivorship bias, problem testing only companies in active.
Look-ahead bias	Use data that it has not been generated during the period analysed.
Data-snooping bias	Overfitting, use parameters to fit the historical data.
Transaction cost	Impact of number of trades and implicit cost.

To begin with, the data sources could contain errors. The dividends are discounted in the stock price when they are produced for that reason, the past data is usually adjusted to not have big drops in prices when, actually, the money is in the account of the investor. Also, the prices have to be adjusted when the company decides to divide a share into multiples shares, the same way the prices have to do.

The reliability of the data has associated a data cost to pay to the provider and must to discount to the final result of the strategy. An important associated cost and typical pitfall to fall not including it in the strategy result, is the transaction cost. This is the price paid to the broker to send the orders to the market. For this reason, the strategies with less trades and great returns in each trade are good to avoid this cost.

The pitfall called survivorship bias is when, for example, the market is analysed from an index where the companies listed are currently in active. The problem comes from not including in the strategy researching the companies that went into bankruptcy at some point because at any moment this could happen and the algorithm is not be prepared for that.

Another important pitfall is the data-snooping bias, occurring when the strategy is well performed with historical data but because the conditions are fitting a particular period of data. To avoid this bias, the strategy has to be test with additional data or “out-of-the-sample” data.

To finalize, it has to be taken into account not use, in a particular point of time in the historical data, information that has not been generated at that point, this is called look-ahead bias. For example, using an annual financial ratio that is brought at the end of the year trying to make operations with it at the begging of the year.

5.4 Strategy performance

In the cycle of the strategy research and even when the algorithm is working in a real environment the performance measures will be the questions to make decisions about the behaviour making profits of a strategy. It is looking for stability, a good risk-reward ratio, rather than high returns.

As an example, two simple strategies are performed: buying two different stocks at the beginning of the period and selling at the end e.g. Amazon and Microsoft:

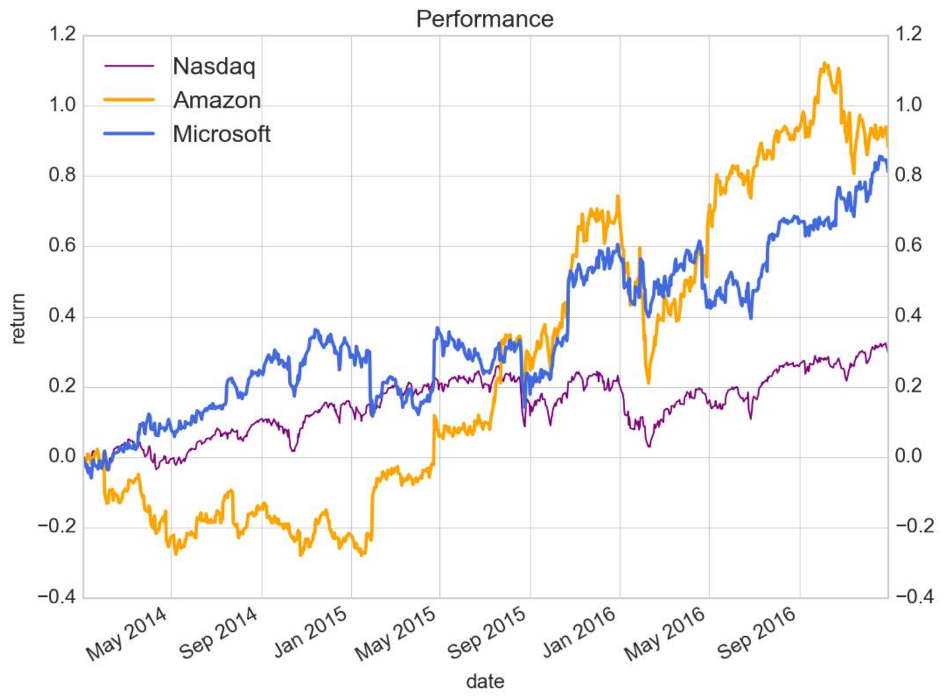


Figure 11 *Performance comparison [Appendix 2]*

Table 5 *Example's Performance [Appendix 2]*

	AMZN	MSFT
Period return	**88.42%**	**81.32%**
Sharpe Ratio	**0.62**	**0.74**
Sortino Ratio	**0.86**	**0.93**
Max drawdown period	21-01-2014 to 23-04-2015 **Days: 457**	28-04-2015 to 22-10-2015 **Days: 177**
Max drawdown	**-30.53%**	**-18.05%**

As it can be seen, Amazon did it better than Microsoft in terms of final returns and even both companies better than the index where they are listed. However, the stability of the strategy shows a different picture.

The Sharpe ratio and Sortino ratio show a better performance risk-reward in Microsoft despite the returns of Amazon. Also, both max drawdown and max drawdown period are better in Microsoft too, which makes the strategy stronger and more stable.

As a general rule, a strategy with a Sharpe Ratio less than 1 is not suitable for an automated algorithm. For a strategy that makes profit every month its Sharpe ratio is usually greater than 2 and for strategies with daily profit it is greater than 3 (Chan, 2008, p. 21).

In this case, both assets are in the Nasdaq Index that got a 30% returns in the same period so the Sharpe ratios tells us that both strategies are good to invests rather than the invest in the index passively.

7 GOALS AND FRAMEWORK DESIGN

In this thesis, a framework will be developed to work from a trading strategy idea to a fully automated algorithm that communicate orders to the market. Therefore, in order to achieve this goal, the framework must provide:

- Gathering real data from the markets.
- Technical indicators to use in trading strategies.
- Back testing tools to test the algorithm designed from the strategies.
- Performance measures of the strategies.
- Integration with a real broker to execute orders given by the trading algorithms.

The framework will be divided in 3 modules to help the design, testing and future changes:

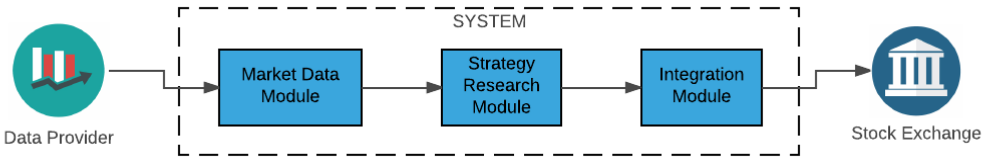


***Figure 12** Framework Design*

7.1 Data Module

The Data Module will be responsible of read, storage and provide market data to of the framework. There are many market data providers for any particular kind of data such as financial data, real time prices, forecasting, news, etc. Generally, the more expensive the data provider is, more quality the data have. In this case, the module will provide just stock prices from a free provider.

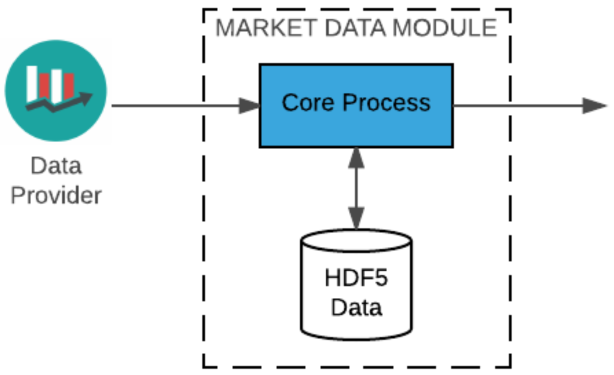


***Figure 13** Market data module design*

The framework feeds on large amounts of data. The strategies could require several months or years of data trough different kind of assets therefore it is needed fast access to this data. The library Pandas provide some methods to storage data in filesystem, for example CSV and HDF5 format:

Writing

```
>>> %timeit -n10 sr.to_csv(os.path.abspath('.\\test.csv'))
10 loops, best of 3: 5.19 s per loop
>>> %timeit -n10 sr.to_hdf(os.path.abspath('.\\test.hdf5'), 'key')
10 loops, best of 3: 22.6 ms per loop
```

Reading

```
>>> %timeit -n10 pd.read_csv(os.path.abspath('.\\test.csv'))
10 loops, best of 3: 369 ms per loop
>>> %timeit -n10 pd.read_hdf(os.path.abspath('.\\test.hdf5'), 'key')
10 loops, best of 3: 11.1 ms per loop
```

As it can be seen, there is a huge difference between a normal csv file and the special hdf5 format for data storage.

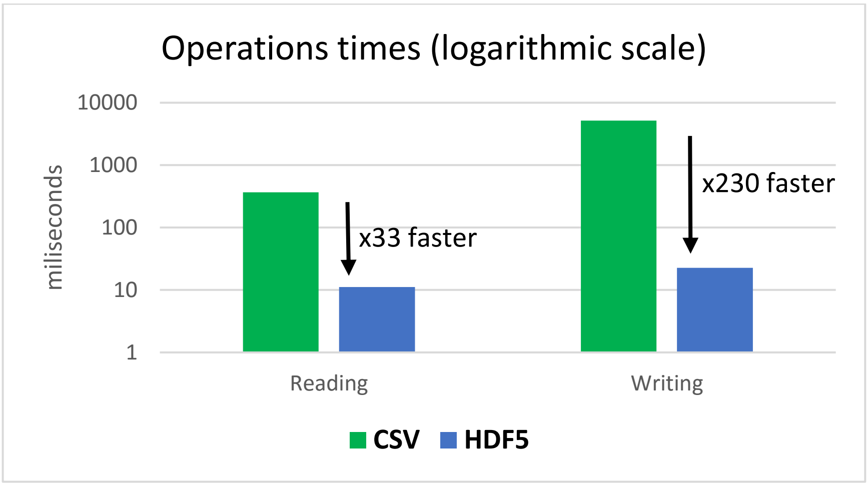


Figure 14 *Format operations comparison*

The Hierarchical Data Format (HDF) is an open source file format used in scientific or business environments which require the use of large sets of data as climate and space research, biotechnology, banking or finance.

The HDF5 uses B-trees to index objects so we can access to the information in complexity O (log n). Furthermore, it is special suitable to work with time series like the financial data to study in this thesis thus this will be the way to storage our information.

7.2 Strategy Research Module

The Strategy Research Module is the most important module of the framework, it will provide the tools to work and research trading ideas such as indicators, performances measures and data visualization. I will be responsible to generate market orders for que integration module.

From the beginning of the process of research trading strategies, the module has the capacity of call the market data module in order to fill the needs of the hosted trading strategies.

It will have a repository of indicators ready to use as well as the respective plotting methods. This repository can be extended with further indicators without affecting the rest of the module or framework.

At the end of each research cycle, the result information will be shown, this performance data will contain ratios, measures and plots to help the research process. The information can be extended for future upgrades as well.

Also, the module will show the trades performed with the historical data as part of performance measures however, the same principle applies to know the trades that have to be communicated to the integration module in order to send them to the market. As it was said at the beginning of the thesis, all the prices are past information, it just a matter of which is the last price and how old they are.

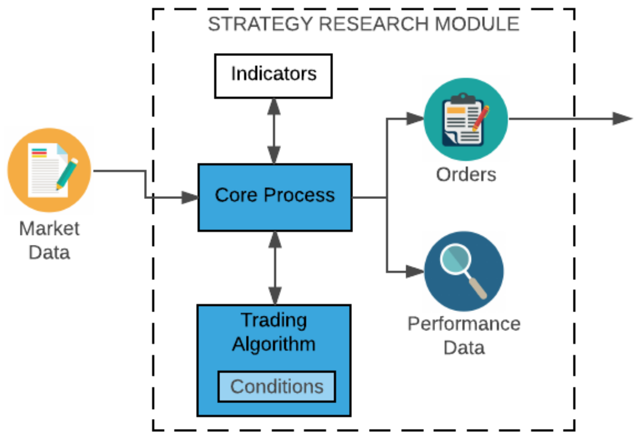


***Figure 15** Strategy Research Module*

7.3 Integration Module

The final step to have a fully automated trading system is a way where the trading strategy communicate the orders to the market, an execution system. As it was said, the communication will be done through the API of a broker.

The broker chosen for this framework will be Interactive Brokers. A well-known broker that provide and API and allow the creation of "demo" accounts to test the result of the transferred orders. This account is ideal to do paper trading.

Unfortunately, the broker's API does not provide Python interfaces to have a direct communication but with the package IBridgePy the module will be able to transfer the orders to the broker. IBridgePy is a wrapper of the C++ Interactive Brokers' API to provide a Python interface (ibridgepy.com, s.f.).

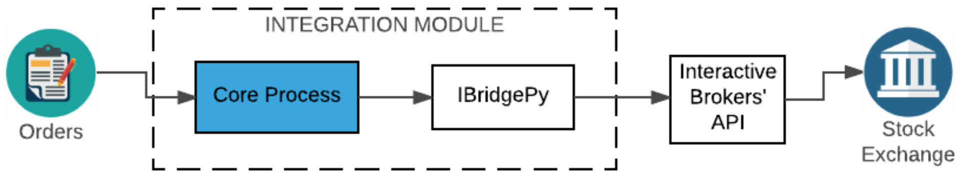


Figure 16 *Integration module and the relations with API*

The process to communicate the orders to the broker through the wrapper IBridgePy has three steps. To begin with, the research module will generate an excel file with the information of the orders to fulfil. This file will contain basic information and is human-readable allowing to modify it in case of need. Also, it will improve the framework modularity.

Table 6 *Orders file composition*

Field	**Description**
Ticket	Acronym that represents the stock in the Broker
Quantity	Number of shares involved in the operation. Negative if it is wanted to sell.
Currency	Currency used in the operation: USD, EUR, etc.

Then, the integration module will read this file and it will create a special one used by IBridge. To finalize, it is necessary execute the special process IBridge has to read the last file created and send the orders to the broker.

The whole process could be executed once a day when the market is closed. At the final of day, the algorithm process the new prices of the market, takes a decision and these decisions are transmitted to the broker for executing it the next day.

8 FRAMEWORK IMPLEMENTATION

8.1 Python motivation

The elements of the system will be developed with the programming language Python. Python is an interpreted language that allows fast code development and live experimentation, convenient for test trading ideas, avoiding the compiling time and increasing the productivity.

The syntax is also close to the mathematical syntax so it is easy to translate financial ratios, formulas or other mathematical concepts into code in a few lines improving the readability.

Python is a well-known language in scientific environments for libraries like NumPy, SciPy, Matplotlib or Pandas that allow mathematical operations over a large amount of data as well as storage and particular functions to read financial data. Also, the plotting of data is incredibly fast and easy.

8.2 Data Module

To begin with, a data provider is needed. Yahoo finance provides free close prices (prices at the end of a market session, daily) and the API is integrated with the Python libraries making convenient for this thesis the use of it. As an example, this is way to get "Adjusted Close" prices (with dividend and splits) of the company "Unilever" represented by the ticket "UN":

```
import market
mrkt = market.Market('yahoo')
prices = mrkt.read_market(start_date = datetime.datetime(2002, 1, 1),
                          end_date = datetime.datetime(2015, 12, 31),
                          ticket = 'UN',
                          column = 'Adj Close')
```

It is necessary to create a "Market" object to start asking for data. This class will set the parameters for storage and default range dates.

Then, it can be called the method `read_market`, responsible to add the data read to a pandas DataFrame. It is easy modifiable to add the functionality to read as many stock ticket as it is wanted to one DataFrame with many prices series but in this case only one ticket can be read per calling.

```python
import pandas_datareader.data as web
import pandas as pd
import datetime

class Market:
    DATA_FILES_PATH = 'c:\\thesis\\data\\'
    START_DATE = datetime.datetime(2000, 1, 1)
    END_DATE = datetime.datetime.today().date()

    def __init__(self, provider):
        self.provider = provider

    def read_market(self, start_date, end_date, ticket, column):
        df = self.read_ticket(start_date, end_date, ticket)
        df = df[[column]]
        new_columns = df.columns.values
        new_columns[0] = 'Prices'
        df.columns = new_columns
        return df
```

The methods `read_ticket` will check the existence in filesystem of the data requested. If it does not exist, initially it will read data from 2000-01-01 until today and it will storage in filesystem in HDF5 format. Depending on the period requested the module will read the data from filesystem and will give only the data in that period. Doing this, the module will request data from internet just one time, maintaining a data buffer in filesystem big enough to attend any petition by the framework for that particular ticket.

```python
    def read_ticket(self,start_date, end_date, ticket):
        ''' Return the ticket data '''
        file = self.DATA_FILES_PATH + ticket + '.hdf5'
        import os.path
        if not os.path.isfile(file):
            msg = 'read_ticket: {} data not found. Start reading from internet'
            print (msg.format(ticket))
            try:
                df = web.DataReader(ticket, self.provider,
                                    self.START_DATE,
                                    self.END_DATE)
                ticket = ticket.replace('^', '_')
                ticket = ticket.replace('.', '_')
                df.to_hdf(file, ticket)
            except Exception as e:
                msg = 'read_ticket: Error reading Internet {} {}'
                print(msg.format(ticket, e))
        else:
            msg = 'read_ticket: {} data already exists'
            print(msg.format(ticket))
            df = pd.read_hdf(file)
        return df[start_date.date().isoformat():end_date.date().isoformat()]
```

8.3 **Indicators**

8.3.1 Simple and Exponential Moving Average (SMA)

Both indicators are shown over the price. It can be seen how the exponential average react faster than the simple average to adapt to the latest prices. Depending on the parameters of the averages (the Windows basically) the prices seems to use the averages as a support to continue trends.

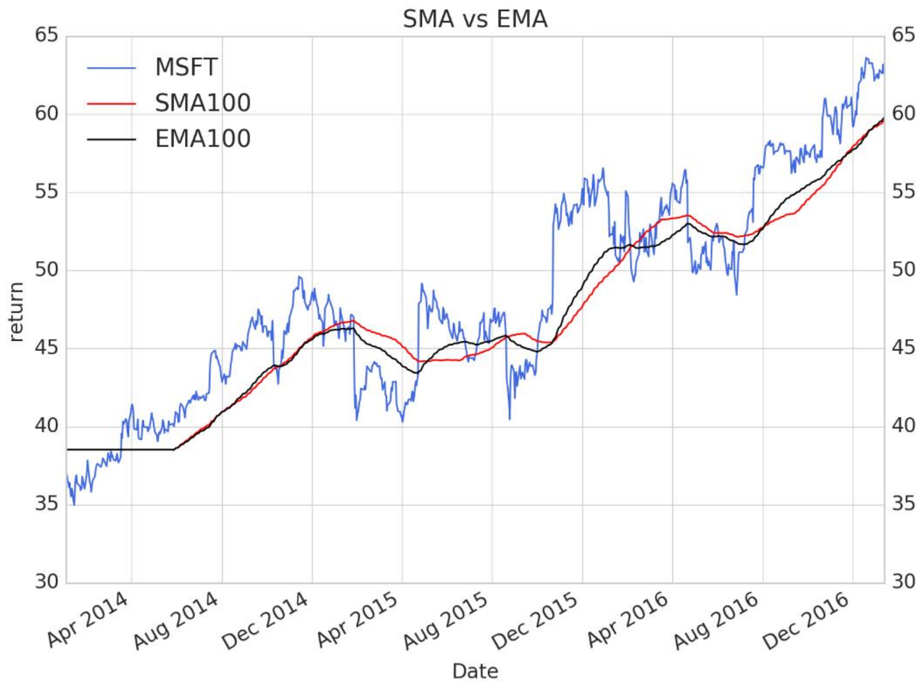


Figure 17 *Comparison SMA vs EMA [Appendix 1]*

The SMA is the simple average of the periods represented in the window. The EMA instead, is calculated with a weight factor that is less and less important while the time goes forward.

```
def SMA(serie, window): ''' Simple Moving Average'''
    sr = pd.Series(np.round(serie.rolling(window).mean(),2))
    sr.name = 'SMA' + str(window)
    return sr

def EMA(serie, window): ''' Exponential Moving Average'''
    multiplier = 2 / (window + 1)
    EMA = pd.Series(index = serie.index)
    #First values
    EMA[window-1] = serie[:window].mean()
    #Rest
    for i in serie.index[window:]:
        prev_d = EMA.index.get_loc(i)-1
        EMA[i] = (serie[i] - EMA[EMA.index[prev_d]])
                  * multiplier
                 + EMA[prev_d]
    EMA[:window] = EMA[window]
    EMA.name = "EMA"+str(window)
    return EMA
```

8.3.2 Relative Strength Index (RSI)

The relative strength index is shown in this plot below the prices to oscillate between a range of 0 and 100. For each day, depending on how the latest returns were, the RSI value will show how the strength of those returns is. The plot shows the typical configuration for the RSI, a window of 14 periods and the limits 70-30 representing levels of oversold and overbought.

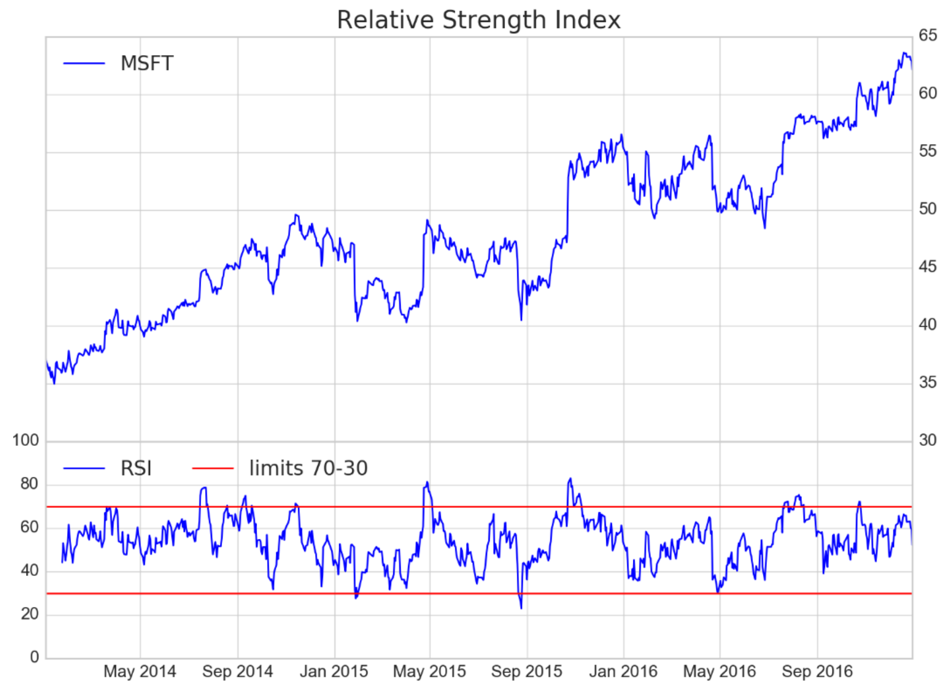


***Figure 18** Microsoft stock and RSI (70,30) [Appendix 1]*

The RSI in calculated as the ratio of a is smoothed exponential averages of gains vs losses in a certain window. This ratio is enclosed between a range from 0 to 100.

```python
def RSI(prices, window=14): '''Relative Strength Index '''
    rsi = pd.DataFrame(prices,index=prices.index)
    # Calculate Gain an Losses
    rsi['diff'] = rsi - rsi.shift(1)
    rsi['pos_diff'] = rsi['diff'][rsi['diff']>0]
    rsi['neg_diff'] = rsi['diff'][rsi['diff']<0]
    rsi['pos_diff'].fillna(0, inplace=True)
    rsi['neg_diff'].fillna(0, inplace=True)
    #Smooth averages
    rsi['avg_gain'] = rsi['pos_diff'].ewm(window).mean()
    rsi['avg_loss'] = rsi['neg_diff'].abs().ewm(window).mean()
    #Null values to the dates without previous
    rsi['avg_gain'][:window] = np.NaN
    rsi['avg_loss'][:window] = np.NaN
    rsi['RS'] = rsi['avg_gain'] / rsi['avg_loss']
    #RS scale 0 - 100
    rsi['RSI'] = 100 - (100 / (1 + rsi['RS']))
    return rsi['RSI']
```

8.3.3 Moving Average Convergence Divergence (MACD)

The Moving Average Convergence Divergence (MACD) is shown in this plot below the prices. It consists of two lines, the black one is the MACD line and the red one the signal line. It can be seen the trend in the prices when the MACD line in above or below the signal line and the changes when both lines cross. Here it is shown the typical configuration 12,26,9 for short, long and signal windows.

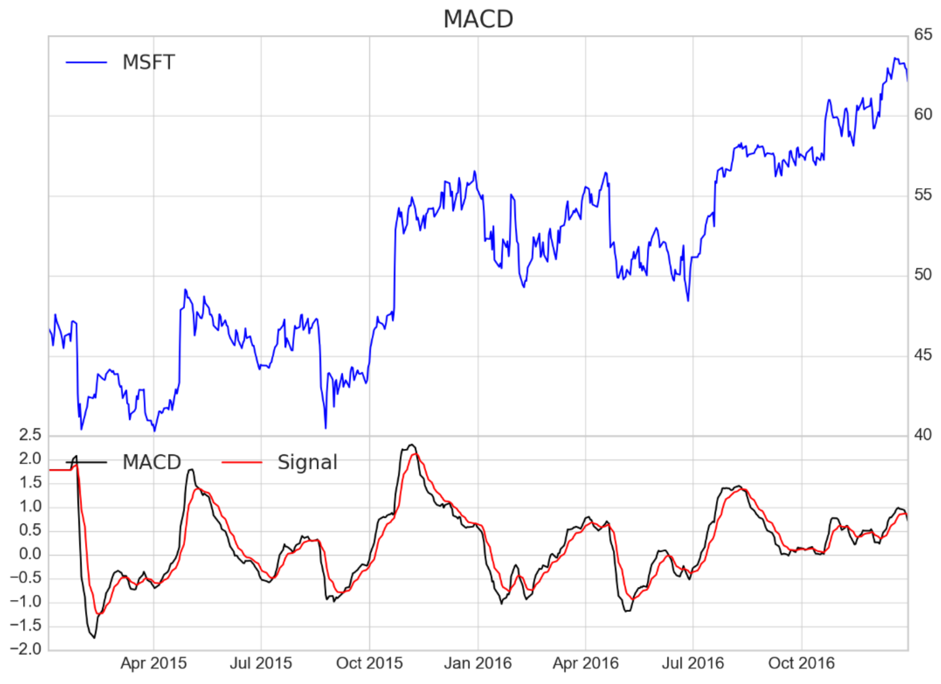


Figure 19 *Microsoft stock and MACD [Appendix 1]*

The MACD line is the difference between a short and a long exponential moving average of the price. The signal line is just an exponential moving average of the MACD line.

```
def MACD(prices, short=12, long=26, signal=9):
    '''Moving Average Convergence Divergence'''
    MACD = pd.DataFrame(index = prices.index)
    MACD['EMAShort'] = EMA(prices, short)
    MACD['EMALong'] = EMA(prices, long)
    MACD['MACD'] = MACD['EMAShort'] - MACD['EMALong']
    MACD['Signal'] = EMA(MACD['MACD'], signal)
    MACD['Hist'] = MACD['MACD'] - MACD['Signal']
    return MACD
```

8.4 Performance information

The performance information will be provided by the following code. It required the time series of the prices of the strategy and the benchmark or index of reference. To begin with, the total return of both strategy and benchmark. It is the percentage of increment respect to the first price of the series.

```
def period_return(prices):
    ''' Return of the whole serie period'''
    return (prices[-1] / prices[0] - 1)
msg = 'Strategy return: {:0.2f}%'
print(msg.format(period_return(strategy) * 100))
msg = 'Benchmark return: {:0.2f}%'
print(msg.format(period_return(index) * 100))
```

The Sharpe ratio it is calculated as the average difference between the daily returns of the strategy and the benchmark divided by the standard deviation of this differences, the risk. Then is multiplied by the square root of 252, that are the trading days in a year, because the unit of the Sharpe Ratio are “per square root time” (Lo, 2002)

```
# Sharpe Ratio: excess return from a benchmark per unit of risk
diff = arithmetic_returns(strategy) - arithmetic_returns(index)
sharpe_ratio = diff.mean() / diff.std() * math.sqrt(252)
print("Sharpe Ratio: {:0.2f}".format(sharpe_ratio))

def arithmetic_returns(prices):
# Dayly Arithmetic returns 15%-15%=0% but it is not true,
# instead use sum of logs
# return serie
return prices / prices.shift(1) - 1
```

For the Sortino ratio it is calculated the same average daily difference for the Sharpe ratio but the risk is the standard deviation of the returns only when are below a desired return. The desired return configurated here is 0% therefore, any positive return will not be penalized as volatility.

```
# Sortino Ratio
diff = arithmetic_returns(strategy) - arithmetic_returns(index)
# Desire return
dr = 0
# TDD: volatility just under the return desired
returns_strg = arithmetic_returns(strategy)
tdd = np.nanstd(np.where(returns_strg>dr,dr,returns_strg))
sortino_ratio = diff.mean() / tdd * math.sqrt(252)
print("Sortino Ratio: {:0.2f}".format(sortino_ratio))
```

To calculate the max drawdown first the difference is calculated between the accumulated prices and the prices. For example, for the Nasdaq:

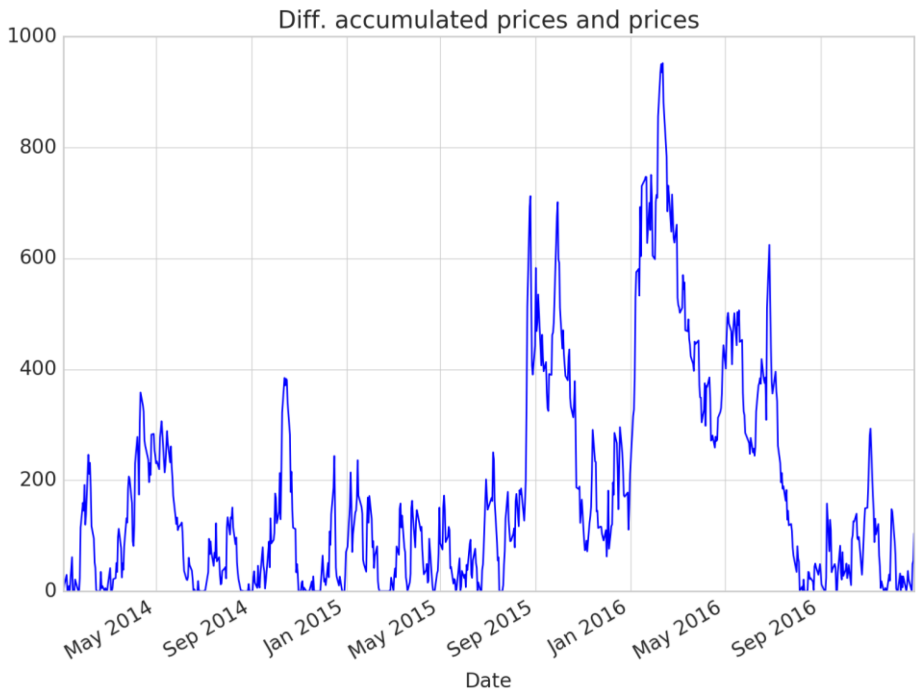


Figure 20 *Difference accumulated prices*

Then, is picked the maximum value getting the lowest moment in the price series, end_period. Until that end_period, is picked the maximum value in the price series as start_period. A simple division gives the maximum drawdown.

```
#Drawdowns
# Difference between accumulated maximum and price
# give me the max drawdown in form of peak, then pick that peak
end_period = np.argmax(np.maximum.accumulate(strategy) - strategy)
# Maximun point before to reach the lowest point of the max drawdown
start_period = np.argmax(strategy[:end_period])
max_drawdown = strategy[end_period] / strategy[start_period] - 1
```

To find the maximum drawdown period, while a new maximum period is found, it is saved to compare it with others until the end of the series is reached. Each drawdown period candidate is calculated getting prices until get one price smaller than the previous ones, indicative of end of drawdowns. Then, it compared the length of the candidate period with the maximum previously saved.

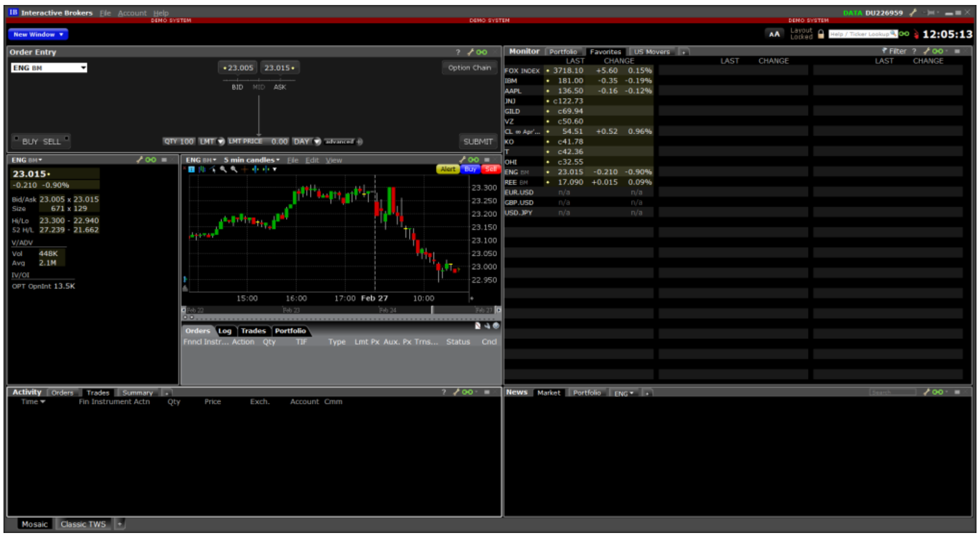
```
max_start_date, max_end_date = max_drawdown_period(strategy)
msg = 'Max drawdown period: {:%d-%m-%Y} to {:%d-%m-%Y}. Days: {}'
print(msg.format(max_start_date, max_end_date,
                 (max_end_date - max_start_date).days))
print('Max drawdown: {:0.2f}%'.format(max_drawdown*100))

def max_drawdown_period(returns):
    start_date = end_date = returns.index[0]
    max_start_date = max_end_date = returns.index[0]
    max_value = 0
    for date, value in returns.iteritems():
        if value > max_value:
            if end_date - start_date > max_end_date - max_start_date:
                max_start_date, max_end_date = start_date, end_date
            max_value = value
            start_date = end_date = date
        else:
            end_date = date
    return max_start_date, max_end_date
```

This algorithm to find the max drawdown period uses a greedy paradigm, as an optimal solution can be found in each stage of the process.

8.5 Integration Module

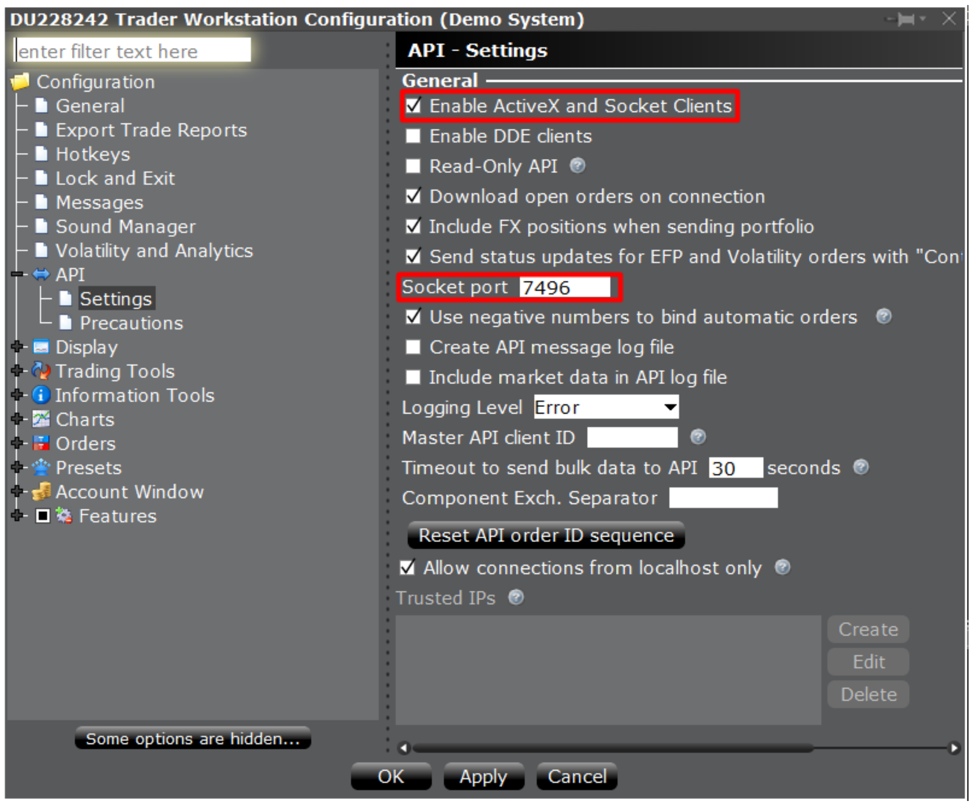
The implementation of the integration module is performed with the broker Interactive Brokers thanks to their API. To test the application, the broker provides a demo account that simulates all the functionality and the trades sent with the stock exchange. This account is especially useful to set the paper trading environment as it takes into account the transaction cost of the trades made by the algorithm. It also provides simulated money to fulfil the orders or, for example, simulate the currency changes when it is necessary.



		LAST	CHANGE	
FOX INDEX	•	3718.10	+5.60	0.15%
IBM	•	181.00	-0.35	-0.19%
AAPL	•	136.50	-0.16	-0.12%
JNJ	•	c122.73		
GILD	•	c69.94		
VZ	•	c50.60		
CL ∞ Apr'...	•	54.51	+0.52	0.96%
KO	•	c41.78		
T	•	c42.36		
OHI	•	c32.55		
ENG BM	•	23.015	-0.210	-0.90%
REE BM	•	17.090	+0.015	0.09%
EUR.USD		n/a		n/a
GBP.USD		n/a		n/a
USD.JPY		n/a		n/a

Figure 21 *Interactive Brokers*

First of all, the “Enable ActiveX and Socket Clients” has to be checked and the port 7494 specified to allow IBridgePy to connect to IB.



DU228242 Trader Workstation Configuration (Demo System)

enter filter text here

- Configuration
  - General
  - Export Trade Reports
  - Hotkeys
  - Lock and Exit
  - Messages
  - Sound Manager
  - Volatility and Analytics
  - API
    - Settings
    - Precautions
  - Display
  - Trading Tools
  - Information Tools
  - Charts
  - Orders
  - Presets
  - Account Window
  - Features

Some options are hidden...

API - Settings

General

- ☑ Enable ActiveX and Socket Clients
- ☐ Enable DDE clients
- ☐ Read-Only API
- ☑ Download open orders on connection
- ☑ Include FX positions when sending portfolio
- ☑ Send status updates for EFP and Volatility orders with "Con

Socket port 7496

- ☑ Use negative numbers to bind automatic orders
- ☐ Create API message log file
- ☐ Include market data in API log file

Logging Level Error

Master API client ID

Timeout to send bulk data to API 30 seconds

Component Exch. Separator

Reset API order ID sequence

- ☑ Allow connections from localhost only

Trusted IPs

Figure 22 *IB API settings*

As an example, the excel orders file is created following the design as the strategy module would do. In this case, one order buying one share of Microsoft.

```
import pandas as pd
# Creating a Buy order to adquire 1 share of Microsft
data = {'Ticket': ['MSFT'], 'Quantity':[1], 'Currency':['USD']}
orders=pd.DataFrame(data, columns=['Ticket','Quantity','Currency'])
# Creating the excel file
orders.to_excel('orders.xlsx')
```

In this implementation, the orders have to be transferred to the broker one by one, but using daily periods this is not a problem as the algorithm will not generate multiple orders per day.

With the orders.xlsx in the filesystem, the integration module can generate the second file for IBridgePy.

```
#Reading the excel file with the algorithm orders
orders = pd.read_excel('orders.xlsx')
# Creating the special file for IBridge
fileName = 'order.py'
fileContent ="""
import time
def initialize(context):
    context.flag=False
#   context.security=symbol('CASH,EUR,USD')
    context.security=symbol('STK,"""+orders['Ticket'][0]+','\
                        +orders['Currency'][0]+"""')
def handle_data(context, data):
    if context.flag==False:
        orderId=order(context.security,""" \
            +str(orders['Quantity'][0])+""")
        order_status_monitor(orderId,
                             target_status='Filled',
                             waitingTimeInSeconds=30)
        context.flag=True
    else:
        time.sleep(10)
        display_all()
        exit()
"""
f = open("../IBridgePy_public_python32/"+fileName,"w")
f.write(fileContent)
f.close()
```

This file contains two methods that IBridgePy will execute. The `initialize` will be executed one time to set the context variables and the `handle_data` many times as it is specified, checking if the order is already in the broker. Then, it is necessary to run the script RUN_ME.py of IBridgePy specifying the file previously generated and the account of Interactive Brokers.

```
#Executing the special file for IBridgePy
fileName='order.py'
accountCode='DU229556' # IB account number
repBarFreq=1 # in seconds
logLevel='INFO'
showTimeZone='US/Eastern'
with open("realMode.txt") as f:
    script = f.read()
exec(script)
```

The result and account information is shown by the script:

```
####    Starting to initialize trader    ####
##    ACCOUNT Balance    ##
CASH=1000249.17
portfolio_value=1000249.17
positions_value=0.0
##    NO ANY POSITION    ##
##    NO OPEN orders    ##
####    Initialize trader COMPLETED    ####
IBridgePy.Trader_single_account::order_quantopian: REQUEST orderId=4
BUY MKT 1 shares of STK,MSFT,USD at unknown price
IBridgePy.Trader_single_account::order_status_monitor: Filled BUY MKT
1 shares of STK,MSFT,USD at 64.11
##    ACCOUNT Balance    ##
CASH=1000249.17
portfolio_value=1000249.17
positions_value=0.0
##    POSITIONS    ##
Symbol Amount Cost_basis Latest_profit
STK,MSFT,USD 1 64.11 NA
##    END    ##
##    OPEN Orders    ##
reqId=4  Filled BUY MKT 1 shares of STK,MSFT,USD at 64.11
##    END    ##
```

Immediately, Interactive Broker shows the order in the tab “Orders”. If the stock exchange of the country where the share is traded, the order will be fulfilled at that moment.

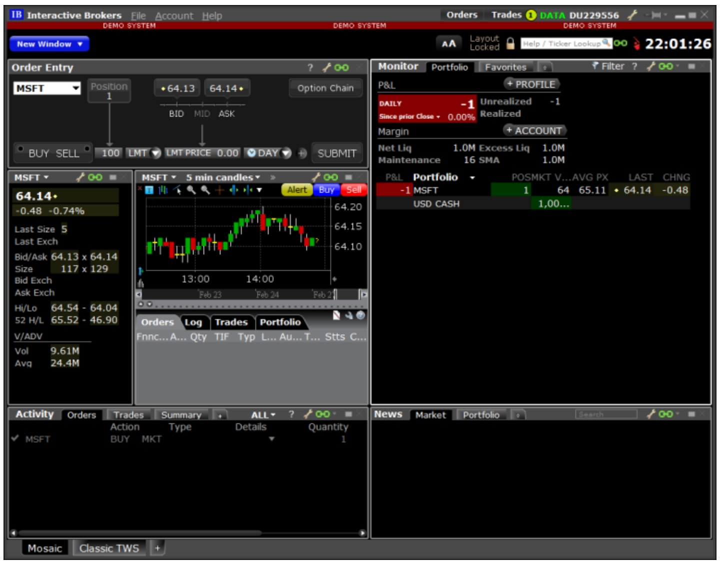


Figure 23 *IB with the sent the order fulfilled.*

This script can be run manually, on demand or with a scheduled task of windows, daily.

9 ALGORITHMIC TRADING STRATEGY

Using the framework developed an algorithmic trading strategy is designed. This section pretends to show the steps performed to the research a strategy and the possible further improvisation of the strategy.

This strategy will be based in two simple moving averages, one “fast” of 42 periods (two months of trading days) and other “slow” of 252 periods (1 year). As it is shown the fast SMA tend to follow the prices closer than the slow SMA and this is useful to identify trends focusing when both SMA cross.

```
# Read market
mrkt = market.Market('yahoo')
nasdaq = mrkt.read_market(start_date = datetime.datetime(2000, 1, 1),
                           end_date = datetime.datetime(2015, 12, 31),
                           ticket = '^IXIC',
                           column = 'Close')
# Creation Moving Average
nasdaq['SMA42'] = indicator.SMA(nasdaq['Prices'], window=42)
nasdaq['SMA252'] = indicator.SMA(nasdaq['Prices'], window=252)
nasdaq['42-252'] = nasdaq['SMA42'] - nasdaq['SMA252']
```

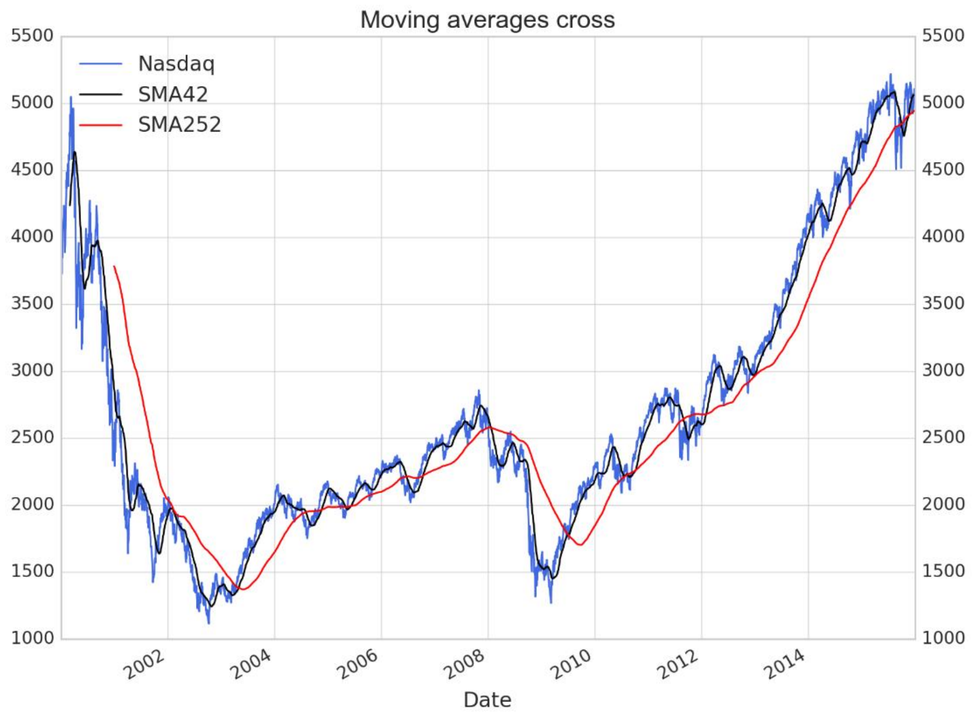


***Figure 24** Nasdaq and fast and slow SMA [Appendix 3]*

The conditions for this strategy will be:

- Buy when the fast SMA cross the slow SMA from below.
- Sell when the fast SMA cross the slow SMA from above.
- Do not trade when both lines are inside a certain threshold of distance.

The last condition prevents to trade when the lines throw a false change in trend, it is looking for a clear trend change.

```
# Threshold
TH = 50
# Buy when short higher than long and greater than threshold
nasdaq['Trades'] = np.where(nasdaq['42-252'] > TH, 1, 0)
# Sell when short lower than long and greater than threshold
nasdaq['Trades'] = np.where(nasdaq['42-252'] < -TH, -1, nasdaq['Trades'])
# Market daily log returns
nasdaq['Market'] = np.log(nasdaq['Prices'] / nasdaq['Prices'].shift(1))
# Constructing strategy, position taken yesterday * today's return
nasdaq['Strategy'] = nasdaq['Trades'].shift(1) * nasdaq['Market']
```

The following plot shows the trading states depending on the previous conditions, 1 for a buy position, -1 for a sell position an 0 when the strategy is out the market.

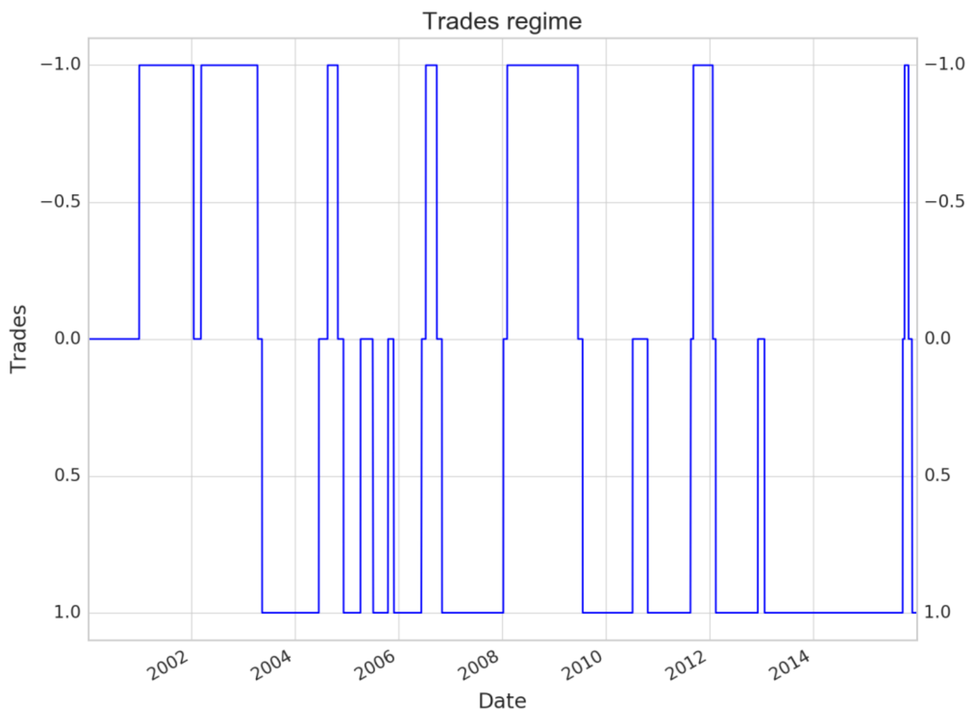


Figure 25 *Trades during the period (-1 sell, 1 buy, 0 out) [Appendix 3]*

The result of those trades are the returns. This strategy tries to follow the trends of the market buying when it is going up and selling when is going down, taking advantage of both sides of the market.

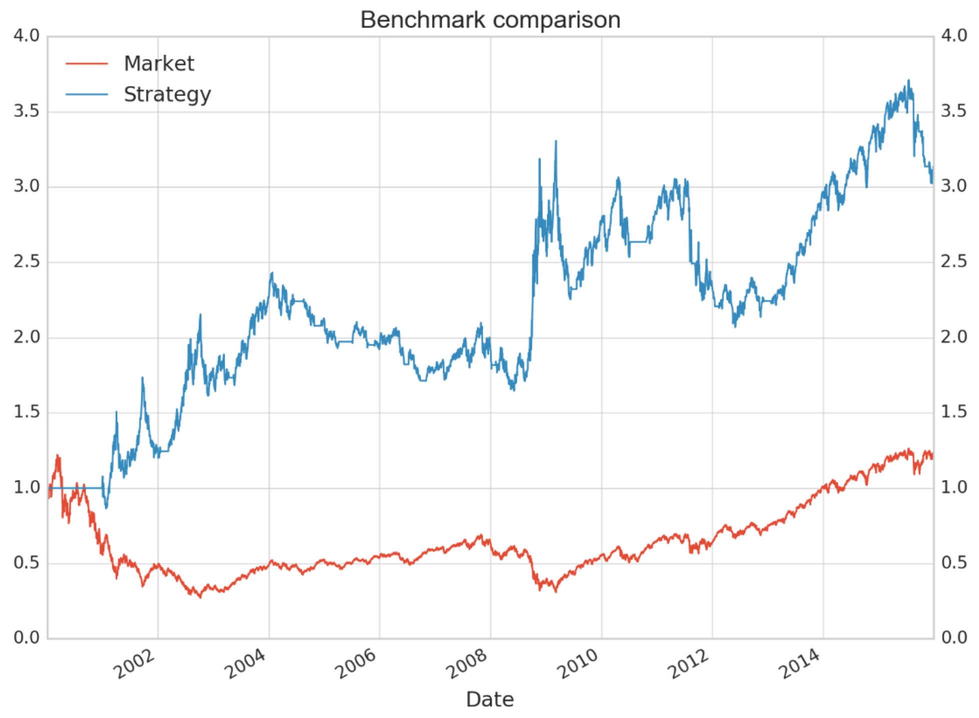


Figure 26 *Strategy result vs index [Appendix 3]*

The final return of the strategy is very high compared with the benchmark but the Sharpe ratio and Sortino ratio show that the volatility makes the strategy not suitable for the real market. Also, the drawdowns are deep and the periods losing money are too long.

```
######### Strategy Performance #########
Strategy return: 207.40%
Benchmark return: 21.21%
Sharpe Ratio: 0.12
Sortino Ratio: 0.34
2012-06-01 00:00:00
Max drawdown period: 09-03-2009 to 07-11-2014. Days: 2069
Max drawdown: -37.45%
#########################################
```

However, thanks to its big returns, the strategy is candidate for risk reduction techniques at the expense of some return. For example, buying the same amount of capital when the strategy is selling and the opposite, of a company or industry more stable therefore the gain and losses will not be as high as before but it will gain stability.

10 CONCLUSION

The framework developed prove the ability to perform the digital operations that algorithmic trading requires. Obtain and storage large amount of market information is fast and efficient. The same technology to process data in other fields like physics simulations or artificial intelligence is used to process financial data.

The construction of algorithmic trading strategies with an interpreted language is convenient due the changes in the strategy conditions and parameters must to be made while the research process, as long as the strategy can be translated in mathematical conditions. Also, the libraries numpy, pandas and matplotlib are a powerful tool to manage market data and plots.

The integration with the market in order to send orders to a broker is possible. However, the technology for an individual investor is not widely extended and just a few brokers have a computer interface to communicate with them.

The measures designed take into account not just the return of the strategy but also the risk involve, to really know if the strategy is worth enough. Also, the plots allow to see what is happening while the algorithm is working.

The problem encountered in this thesis was to find profitable trading strategies over the risk taken. It was learnt that the profitable strategies already known by the investors push the market to a new level of equilibrium because to exploit inefficiencies make them disappear. It required a constant research of new ideas.

During the developing of this thesis it was acquired knowledge from both technological and financial point of view, being useful to better understand processes and systems involve in a financial company.

Starting from the bases settled, the algorithmic trading is a subject big enough to explore, for example, research long-term strategies based in fundamental data or high frequency trading. But also, the study of operations not directly related to make profit like algorithms aimed to reduce the market impact when it is wanted to trade a large number of shares, weighting companies in a portfolio the simulate an index or risk management.

The finance technology will continue evolving along with the algorithmic trading.

REFERENCES

Brito , J., & Castillo, A. (2013). Retrieved from BITCOIN A Primer for Policymakers: https://www.mercatus.org/system/files/Brito_BitcoinPrimer.pdf

Buffet, W. (1984). *businessinsider.* Retrieved from http://www.businessinsider.com/warren-buffett-on-efficient-market-hypothesis-2010-12?r=US&IR=T&IR=T

Chan, E. (2008). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business.* John Wiley & Sons .

de Vries, J., & van der Woude, A. (1997). *The First Modern Economy: Success, Failure, and Perseverance of the Dutch Economy.* Cambridge University Press.

Dickson. (1960). *The Sun Insurance Office 1710–1960: The History of Two and a half Centuries of British Insurance.* London: Oxford University Press.

Fama, E. F. (1965). Retrieved from Random Walks in Stock Market Prices: https://www.chicagobooth.edu/~/media/34F68FFD9CC04EF1A76901F6C61C0A76.PDF

Fama, E. F. (1970). Retrieved from EFFICIENT CAPITAL MARKETS: A REVIEW OF THEORY AND EMPIRICAL WORK: http://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1970.tb00518.x/full

FT. (2016). *Financial services companies braced for automation.* Retrieved from Financial Times: https://www.ft.com/content/577216d8-0802-11e6-a623-b84d06a39ec2

Glantz, M., & Kissell, R. (n.d.). Multi-Asset Risk Modeling: Techniques for a Global Economy in an Electronic and Algorithmic Trading Era. 258. Retrieved from Academic Press: https://books.google.com/books?id=7TcTAAAAQBAJ&pg=PA258&lpg=PA258&dq=algorithmic+trading,+percent+of+total+trades+%22Aite+Group%22&source=bl&ots=4UYXH8tFS8&sig=IDlUyklg7qVjGGo7X9wk6o4RGdc&hl=en&sa=X&ved=0CCUQ6AEwAWoVChMIhfuFhKToyAIVA2MmCh0YDwZj#v=onepage&

Hayek, F. (1945). *The Use of Knowledge in Society.* Retrieved from http://www.econlib.org/library/Essays/hykKnw1.html

ibridgepy.com. (n.d.). *About IBridgePy*. Retrieved from ibridgepy: http://www.ibridgepy.com/introduction/

IDC. (n.d.). Retrieved from http://www.idc.com/getdoc.jsp?containerId=prUS41216616

Koehler, C. (2011, May 31). Retrieved from The Relationship between the Complexity of Financial Derivatives and Systemic Risk: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2511541

Lin, T. C. (2012). Retrieved from A Behavioral Framework for Securities Risk: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2040946&download=yes

Lin, T. C. (2014, March 30). *The New Financial Industry.* Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2417988

Lo, A. W. (2002). *edge-fund.* Retrieved from edge-fund: The Statistics of Sharpe Ratios

Macesich, G. (2000). *Issues in Money and Banking.* Greenwood Publishing Group. Retrieved from https://books.google.com/books?id=k1OYMZ8OzMUC&pg=PA42

Malmendier, U. (2008, December). Retrieved from Law and Finance “at the Origin”: http://eml.berkeley.edu//~ulrike/Papers/JELDraft73_withabstract_andTable.pdf

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets A Comprehensive Guide to Trading Methods and Applications.* New York: New York Institute of Finance.

Pareek, M. (2008). *Clarifying The Information Ratio And Sharpe Ratio*. Retrieved from Seeking Alpha: http://seekingalpha.com/article/63911-clarifying-the-information-ratio-and-sharpe-ratio

Red Rock. (n.d.). *Sortino: A ‘Sharper’ Ratio.* Retrieved from Red Rock Capital: http://www.redrockcapital.com/Sortino__A__Sharper__Ratio_Red_Rock_Capital.pdf

Sharpe, W. F. (1994). Retrieved from The Sharpe Ratio: http://web.stanford.edu/~wfsharpe/art/sr/sr.htm

StockChart.com. (n.d.). *Relative Strength Index (RSI).* Retrieved from http://stockcharts.com: http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages

Tulchinsky, I. (2015). *Finding Alphas: A quantitative approach to building trading strategies.* Wiley.

Watson, P. (2005). *Ideas: A History of Thought and Invention from Fire to Freud.* New York: HarperCollins Publishers.

Wyss, B. O. (2000). *Fundamentals of the Stock Market.* McGraw-Hill.

YaleNews. (2011). Retrieved from http://news.yale.edu/2011/02/17/shiller-paper-cited-one-century-s-top-economic-articles

Appendix 1

PLOTS CODE

Random walk plot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

plt.style.use('seaborn-whitegrid')

n = 100
for i in range(7):
    pd.Series(np.random.random_integers(-1,1, size=n).cumsum()).plot()
plt.savefig('random_walk.png', dpi=200)
```

Example_SMAvsEMA

```
import market
import matplotlib.pyplot as plt
import datetime
import indicator

# Read prices
mrkt = market.Market('yahoo')
column = 'Close'
prices = mrkt.read_market(start_date = datetime.datetime(2014, 1, 1),
                          end_date = datetime.datetime(2017, 2, 16),
                          ticket = 'MSFT',
                          column = column)
# Costruction indicators from prices
SMA = indicator.SMA(prices[column], 100)
EMA = indicator.EMA(prices[column], 100)

# Plotting
plt.style.use('seaborn-whitegrid')
prices[column].plot(color='royalblue', label='MSFT')
SMA.plot(color='red')
EMA.plot(color='black')
plt.tick_params(axis='y', which='both', labelright='on')
plt.legend(loc='best')
plt.ylabel('return')
plt.title("SMA vs EMA")
plt.tight_layout()

plt.savefig('./figures/example_SMAvsEMA.png', dpi=200)
```

Example_RSI

```python
import market
import matplotlib.pyplot as plt
import datetime
import indicator

column = 'Close'
mrkt = market.Market('yahoo')
prices = mrkt.read_market(start_date = datetime.datetime(2014, 1, 1),
                          end_date = datetime.datetime(2016, 12, 31),
                          ticket = 'MSFT',
                          column = column)

# Construct RSI data form prices
rsi = indicator.RSI(prices[column], 14)

# Figure parameters
left, width = 0.05, 0.9
rect1 = [left, 0.35, width, 0.56]
rect2 = [left, 0.05, width, 0.3]

# Plotting
plt.style.use('seaborn-whitegrid')
fig = plt.figure()
ax1 = fig.add_axes(rect1)
ax2 = fig.add_axes(rect2, sharex=ax1)
ax1.plot(prices[column], label='MSFT', color = "blue")
ax2.plot(rsi, color= 'blue')
ax2.axhline(70, color='r', label='limits 70-30')
ax2.axhline(30, color='r')
ax1.legend(loc='upper left')
ax2.legend(loc='upper left', ncol=2)
ax2.set_ylim([0, 100])
ax1.set_title('Relative Strength Index')
ax1.tick_params(labelleft='off', labelright='on')

plt.savefig('./figures/RSI_example.png', dpi=200)
```

Example_MACD

```python
import market
import matplotlib.pyplot as plt
import datetime
import indicator
import pandas as pd

mrkt = market.Market('yahoo')
column = 'Close'
prices = mrkt.read_market(start_date = datetime.datetime(2015, 1, 1),
                          end_date = datetime.datetime(2016, 12, 31),
                          ticket = 'MSFT',
                          column = column)

# Constructing indicator from prices
MACD = indicator.MACD(prices[column], 12, 26, 9)

# Figure parameters
left, width = 0.05, 0.9
rect1 = [left, 0.35, width, 0.56]
rect2 = [left, 0.05, width, 0.3]

# Plotting
plt.style.use('seaborn-whitegrid')
fig = plt.figure()
ax1 = fig.add_axes(rect1)
ax2 = fig.add_axes(rect2, sharex=ax1)
ax1.plot(prices[column], label='MSFT', color = "blue")
ax2.plot(MACD['MACD'], color= 'black')
ax2.plot(MACD['Signal'], color= 'red')
ax1.legend(loc='upper left')
ax2.legend(loc='upper left', ncol=2)
ax1.set_title('MACD')
ax1.tick_params(labelleft='off', labelright='on')

plt.savefig('./figures/example_MACD.png', dpi=200)
```

Appendix 2

SUPPORT CODE

Microsoft – Amazon Strategy performance

```
import matplotlib.pyplot as plt
import market
import datetime
import util

# Read market
mrkt = market.Market('yahoo')
column = 'Adj Close'

nasdaq = mrkt.read_market(start_date = datetime.datetime(2014, 1, 1),
                          end_date = datetime.datetime(2016, 12, 31),
                          ticket = '^IXIC',
                          column = column)

msft = mrkt.read_market(start_date = datetime.datetime(2014, 1, 1),
                            end_date = datetime.datetime(2016, 12, 31),
                            ticket = 'MSFT',
                            column = column)

amzn = mrkt.read_market(start_date = datetime.datetime(2014, 1, 1),
                            end_date = datetime.datetime(2016, 12, 31),
                            ticket = 'AMZN',
                            column = column)

 # Plotting
plt.style.use('seaborn-whitegrid')

returns_day = util.normalize(nasdaq['Prices'])
returns_day.plot(label='Nasdaq', c = 'purple', lw = 1)

returns_day = util.normalize(amzn['Prices'])
util.info(amzn['Prices'], nasdaq['Prices'])
returns_day.plot(label='Amazon', c = 'orange', lw = 2)

returns_day = util.normalize(msft['Prices'])
util.info(msft['Prices'], nasdaq['Prices'])
returns_day.plot(label='Microsoft', color = 'royalblue', lw = 2)

plt.title('Performance')
plt.grid(True)
plt.legend(loc='best')
plt.ylabel('return')
plt.xlabel('date')
plt.tick_params(axis='y', which='both', labelright='on')
plt.tight_layout()
plt.savefig('./figures/perf_impl.png', dpi=200)
```

STRATEGY CODE

Momentum Strategy

```python
import market
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
import indicator

mrkt = market.Market('yahoo')
nasdaq = mrkt.read_market(start_date = datetime.datetime(2000, 1, 1),
                          end_date = datetime.datetime(2015, 12, 31),
                          ticket = '^IXIC',
                          column = 'Close')
# Creation Moving Average
nasdaq['SMA42'] = indicator.SMA(nasdaq['Prices'], window=42)
nasdaq['SMA252'] = indicator.SMA(nasdaq['Prices'], window=252)
nasdaq['42-252'] = nasdaq['SMA42'] - nasdaq['SMA252']

# Threshold
TH = 50
# Buy when short higher than long and greater than threshold
nasdaq['Trades'] = np.where(nasdaq['42-252'] > TH, 1, 0)
# Sell when short lower than long and greater than threshold
nasdaq['Trades'] = np.where(nasdaq['42-252'] < -TH, -1, nasdaq['Trades'])
# Market daily log returns
nasdaq['Market'] = np.log(nasdaq['Prices'] / nasdaq['Prices'].shift(1))
# Constructing strategy, position taken yesterday * today's return
nasdaq['Strategy'] = nasdaq['Trades'].shift(1) * nasdaq['Market']

# Plotting
plt.style.use('seaborn-whitegrid')
# Cumsum of daily log retunrns and apply exp
nasdaq[['Market','Strategy']].cumsum().apply(np.exp).plot()
plt.title('Benchmark comparison')
plt.tick_params( labelright='on')
plt.tight_layout()
plt.savefig('./figures/methodology_1.png', dpi=200)

plt.figure()
plt.ylim([1.1,-1.1])
nasdaq['Trades'].plot(c='blue')
plt.title('Trades regime')
plt.ylabel("Trades")
plt.tick_params(labelright='on')
plt.tight_layout()
plt.savefig('./figures/methodology_2.png', dpi=200)

plt.figure()
nasdaq['Prices'].plot(label='Nasdaq', color='royalblue')
nasdaq['SMA42'].plot(color='black')
nasdaq['SMA252'].plot(color='red')
plt.legend(loc='upper left')
plt.title('Moving averages cross')
plt.tick_params(labelright='on')
plt.tight_layout()
plt.savefig('./figures/methodology_3.png', dpi=200)
```