AUTOMATED BITCOIN TRADER

INTRODUCTION

This EPQ aims to investigate the Bitcoin trading market, and produce an automated trader that is capable of automatically trading Bitcoins for a fiat currency, based on market trends.

WHAT IS BITCOIN?

Bitcoin is a decentralised/peer-to-peer[1] crypto-currency[2] that began circulation in 2009. This means no party or bank is in charge of it. Satoshi Nakamoto published a paper[3] in late 2008, describing an abstract concept of how Bitcoin could work. He came up with the idea after identifying the main weaknesses in online commerce – trust is required, especially due to the ability to reverse transactions. This usually meant that a third party (i.e. a bank) is required to mediate the transaction, in order to prevent fraud. The entire system is based on trust. What if transactions were based on something other than trust?

Bitcoin is an "an electronic payment system based on cryptographic[4] proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party"

TRADING BITCOIN

One method of acquiring Bitcoin is by purchasing it online. The price of Bitcoin is variable, as one Bitcoin will never (intentionally) be the same value as one Bitcoin at any other instance of time. This is the reverse for fiat currencies, such as the pound sterling, as one pound is always one pound at any instance of time in Britain. Thus, the crypto-currency has become a tradable commodity on various online exchanges, and can be traded in a similar manner as on a foreign exchange[5], apart from the far lower trade frequency and lower liquidity[6] in the Bitcoin market.

WHY I CHOSE THIS PROJECT

I have a strong passion for software engineering, and on the technical side, it provided me with an opportunity to learn and apply new and exciting technologies that were fit for the purpose. I was eager to combine this with some form of economics. I was generally interested in trading, and Bitcoin was a currency that I had previous exposure. The lack of high frequency/professional traders in the market allows anyone to explore and experiment with both the Bitcoin world, and trading in general. Also, the Bitcoin markets are much more volatile, providing for some interesting opportunities for profit.

[1] See appendix 1.1
[2] See appendix 1.2
[3] Bitcoin: An Electronic Peer-to-Peer Cash System – See bibliography
[4] See appendix 1.3
[5] See appendix 1.4
[6] See appendix 1.5

TRADING

DATA FORMAT

To appreciate the following strategies, it is important to know understand how prices are represented. Often, charts will refer to candlesticks[7], as opposed to a line graph of prices. A candlestick will explain what the first (opening), last (closing), highest and lowest trade prices are for a given interval (e.g. for that hour). A series of candlesticks are then displayed, such that the price data makes sense. There are 2 types of candlesticks: a hollow candlestick indicates that the closing price is higher than the opening price (i.e. the price of Bitcoin has increased at the end of that period). A filled in candlestick represents the exact opposite. Note that the values of the open and close have been switched around for the respective candlestick.

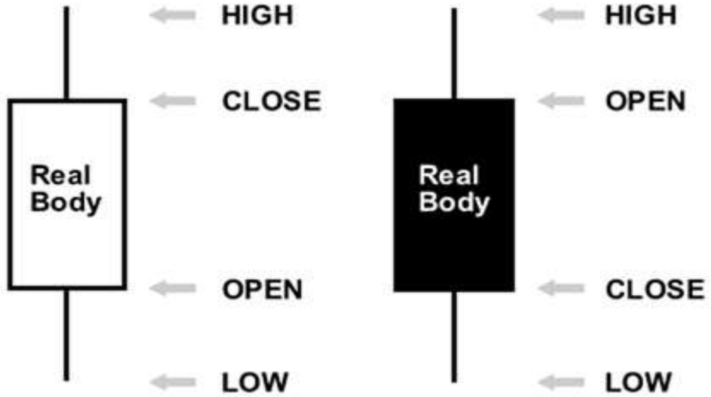


Figure 1. Hollow and Filled Candlesticks - http://knownware.com/wp-content/uploads/2012/11/Candlestick-Example.png

The process of turning many rows of data from the form of "[trade at price, time]" into corresponding [time at interval of X, open trade price, high trade price, low trade price, close trade price]" candlestick form will be discussed further on in this document. This is something I made the mistake of not doing, and hence my results did not make sense since they were all over variable time periods (the difference in time between two trades happening are entirely random)

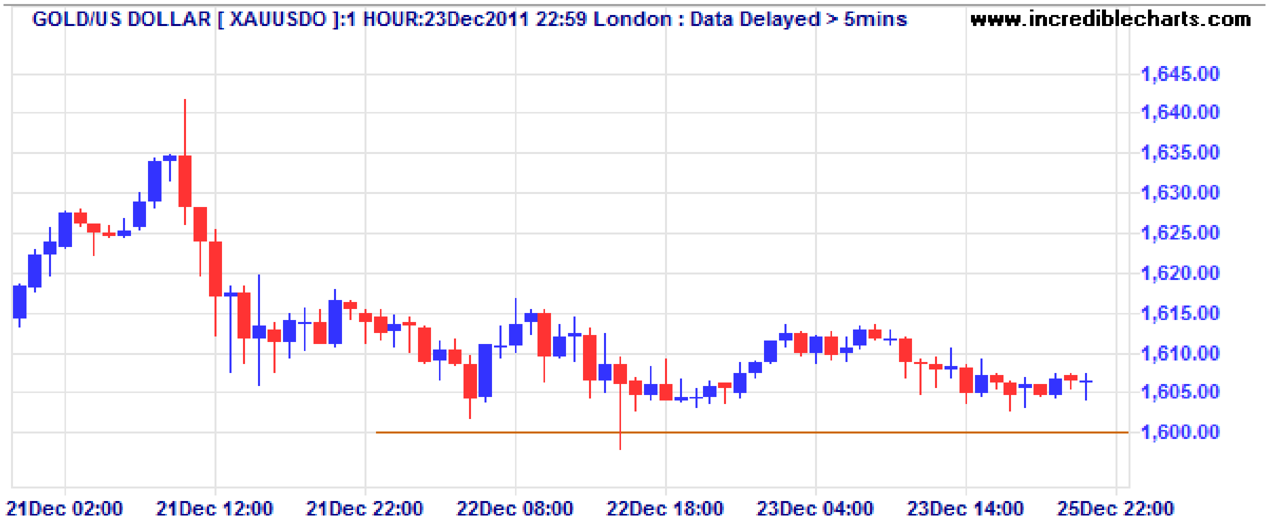


Figure 2. Gold/USD graph. A candlestick represents a time interval of 1 hour. The blue represents hollow candlesticks, and the red represents filled candlesticks. - http://static.incrediblecharts.com/images/2011/2011-12-24-gold.png

[7] See appendix 1.10

MOVING AVERAGE BASED STRATEGIES

SMA CROSSOVER – SIMPLE MOVING AVERAGE CROSSOVER

A simple moving average (SMA) is an average that slides as more data arrives. It takes two parameters: a list of prices with most recent at the top, and the period over which the SMA should be calculated. The formula is essentially the same a normal average.

$$SMA(interval, values) = \frac{values_1 + \ values_2 + \ldots \ values_{interval}}{interval}$$

This is calculated every time a new value is added to the top of the list, and thus the average keeps moving along with the data.

One can take 2 SMAs of different time period, where one is a long term SMA and the other a short term SMA.

The long term SMA represents and reacts to long term and significant changes in the market, and the short term SMA reacts to short term changes in the market, since the calculation uses fewer values.
Plotting these two lines on the same graph can then be used to trigger certain actions:

- A bearish[8] market is indicated when the long term SMA crosses (is greater than) the short term one, so one should sell all Bitcoins back for the fiat currency, as the local maximum has been reached, and the price is likely to head downwards.

- A bullish[9] market is represented by the reverse situation; the short term SMA crosses (is greater than) the long term SMA. Bitcoin should be bought and stored until the market shows signs of becoming bearish again.

The SMA was included as the concept is the basis for all moving average-based strategies. A clear comprehension of the SMA ensures the further strategies are easy to understand, as many are crossover based strategies.

There is a major disadvantage to all moving average strategies: lag[10] - the reaction time to a change in trend. The SMA has the largest lag, since it does no more than average data.

I do not think the SMA crossover is a practical indicator[11] in a fully automated environment – it hardly reacts quickly enough to huge market changes to be solely effective. It perhaps could be used in a semi-automatic mode, where the trades are not carried out without human approval, but offers no advantages over the better strategies we are about to discuss.

[8] See appendix 1.15
[9] See appendix 1.14
[10] See appendix 1.10
[11] See appendix 1.16

EMA CROSSOVER – EXPONENTIAL MOVING AVERAGE CROSSOVER

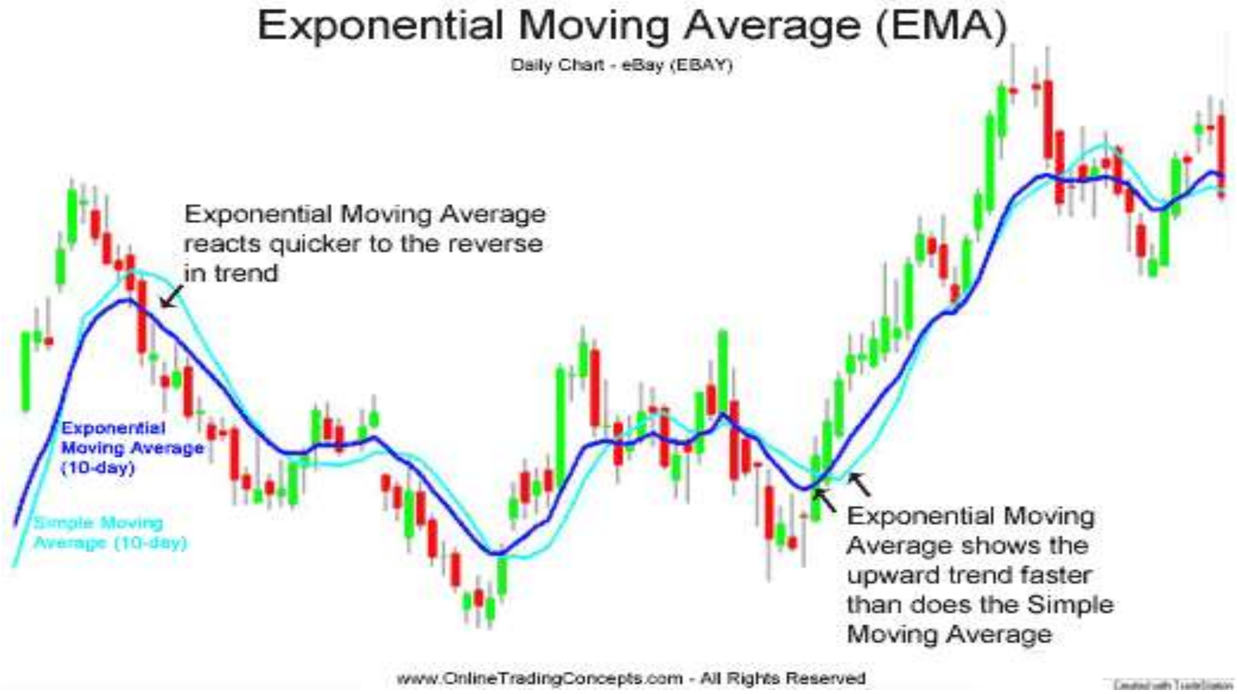
This is very similar to a simple moving average, except that the items are multiplied by an exponential weight factor depending on how recent the item is. This means the average reacts more quickly to recent data and allows a quicker prediction of market trend. If the data is more recent, the importance it is given is exponentially larger than the others.

$$EMA(interval, values) = \frac{(values_1 + \alpha values_2 + \alpha^2 values_3 + \ldots \alpha^{interval-1} values_{interval})}{{}^{1}/_{(1-\alpha)}}$$

Where $\alpha = \ (1 - \frac{2}{N+1})$ and thus $\alpha < 1$

Although an understanding of the formula or what the constant α is not required, note that as a value is less recent, the size of the weighting factor decreases exponentially from 1 downwards. This makes the EMA react much more quickly to a change in trend.

The same crossover strategy as SMA is applied, instead using 2 EMA lines. A graph depicting the difference in the effectiveness of these 2 strategies is below:



Exponential Moving Average reacts quicker to the reverse in trend

Exponential Moving Average shows the upward trend faster than does the Simple Moving Average

www.OnlineTradingConcepts.com - All Rights Reserved

Figure 3. SMA vs EMA. The EMA is reacting more appropriately to the trend than the SMA, although there is huge lag. http://www.onlinetradingconcepts.com/images/technicalanalysis/MAexponentialEBAY.gif

This method is very well documented, and is a core indicator that the Bitcoin community use. This made it easy to ensure that I was calculating the correct values, as I was able to cross-reference results online. The EMA can be on different time intervals and of different lengths, and the ability to just these parameters makes it extremely flexible. For example, the 21 day EMA against the 11 day EMA over a daily interval is an extremely common (and therefore fruitless) configuration.

I chose the EMA, as it was naturally a step up from the SMA, and is practical to fully automate in an automated environment. The lag is better handled, but we can go a step further.

DEMA/TEMA CROSSOVER – DOUBLE/TRIBLE MOVING AVERAGE CROSSOVER

These are variants of the EMA, and are designed to remove as much lag as possible from market data. This is done by applying even more weight to the more recent data, resulting in a more reactive indicator.

The DEMA of a certain value is given by:

$$DEMA = 2 * EMA - EMA(EMA)$$

Clearly, more weight will be applied if the value is first in the list, as the EMA will be larger.

Similarly, the TEMA is given by:

$$TEMA = (3 * EMA - 3 * EMA(EMA)) \; + \; EMA(EMA(EMA))$$

These are all applied in the same way as the SMA and EMA. They have been chosen, as they remove much of the lag associated with the EMA, however are not always the best choice, as they overshoot and undershoot more. This can result in bad buy/sell signals, as they are now vulnerable to more market noise[12], and thus the usage of the DEMA and TEMA are subject to the conditions of the market, and the user's personal preference.

The graph below highlights the difference between the three strategies:

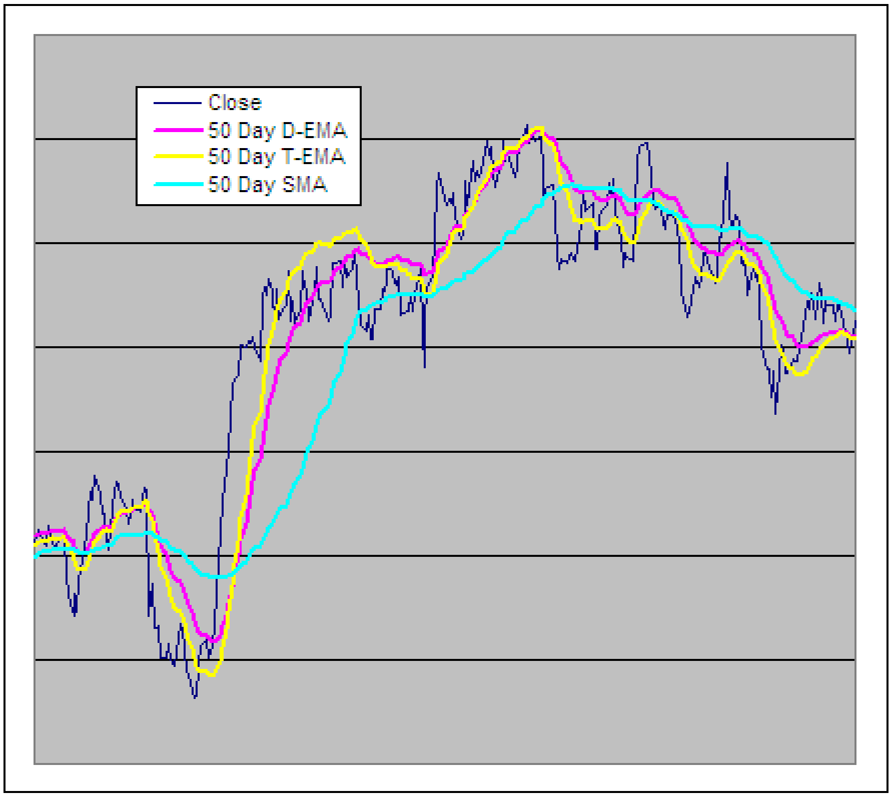


Figure 4. SMA vs TEMA vs DEMA. Close indicates the closing price of a would-be candlestick. The TEMA fits the trend the best, but can indicate non-profitable signals. http://etfhq.com/blog/2010/11/17/double-and-triple-exponential-moving-average/

[12] See appendix 1.17

(K)AMA – (KAUFMANN'S) ADAPTIVE MOVING AVERAGE[13]

An adaptive moving is a type of moving average that adapts, based on market momentum. In genuine trends, the AMA shifts in the direction of the trend.

In noisy periods, where the price is fluctuating, the AMA is more consistent. Therefore, the market noise is filtered out, and a better indicator of when a trend will start is produced.

This basic idea is to incorporate a volatility ratio into the EMA, and use this as a smoothing factor. As a significant change in trend occurs, the weighting increase to 1 or -1. As smaller fluctuations occur, the weighting tends to 0, thus giving the values hardly any relevance.

The full calculation is too complex for inclusion in this document, however can be found in the by following the link in bibliography, under "Adaptive Moving Average".

A trade would occur if the AMA shifts in the opposite direction. For example, if the AMA was trending positively, and then changed direction, it is indicating that one should sell Bitcoins, and vice-versa.

I included the AMA because it is an uncommon strategy that poses several advantages over the others. The volatility ratio could produce more stable and consistent results in the Bitcoin market, albeit with less short-term profit and less risk.

A graph of the SMA vs the AMA is below, and you can see that it enters the trend more slowly, but is much flatter during noisy periods (on the right):

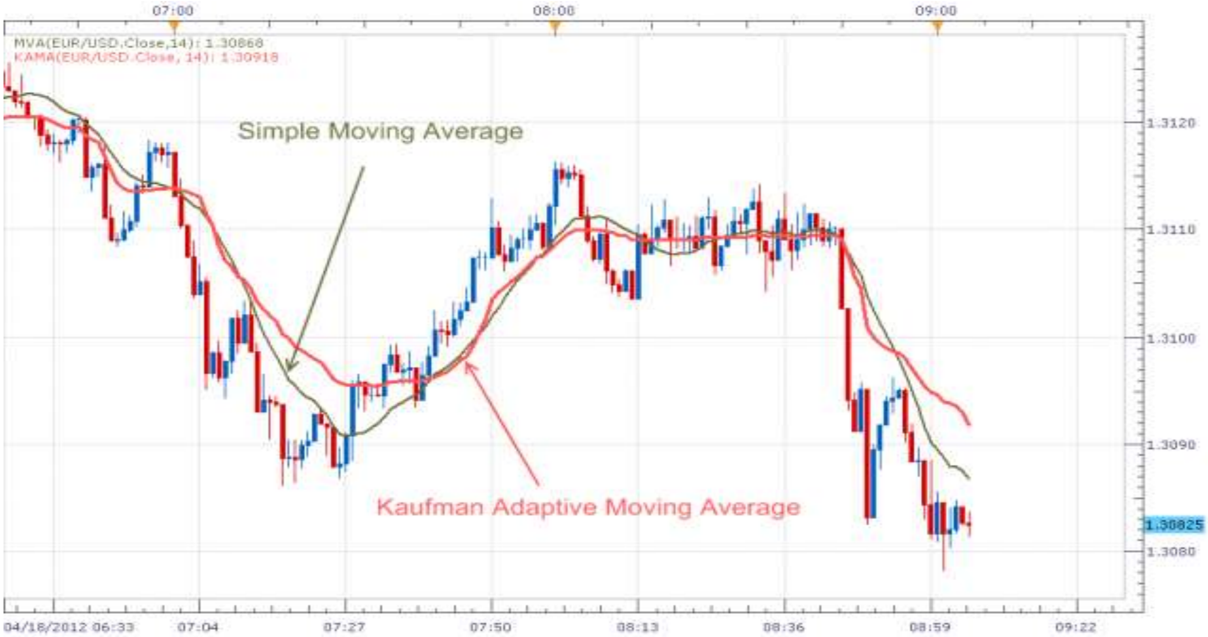


Figure 5. SMA vs AMA. http://fxcodebase.com/wiki/images/thumb/d/d7/KAMA1.png/640px-KAMA1.png

I considered applying the idea of the AMA, using the DEMA or TEMA instead. This could mean the average reacts a lot faster to a change in trend, however the implementation is much harder.

[13] Adaptive Moving Average – See bibliography link for in-depth explanation

MACD – MOVING AVERAGE CONVERGENCE DIVERGENCE

This strategy is based on the ideas of the EMA. A signal line, which is (usually) a 9-day EMA is plotted. The actual MACD is typically calculated and plotted by subtracting the 26-day EMA from a 12-day EMA. The MACD is a formal name for a type of strategy that uses three EMAs.

All analysis using the MACD is relative to the signal line, and so it is usually plotted on its own graph, underneath the price data for clarity.

When the MACD crosses above the signal line, this indicates a bullish signal, hinting that it may be a good time to buy. Conversely, if the MACD crosses below the signal line, it may be time to sell as it is possible that we are entering a bearish market.

Like the EMA crossover, one must be wary of crossover being fraud, in the sense that it is minor and likely to flip back the other way.

When the price diverges from the MACD, the current trend has likely ended. This may mean the trader should reconsider their strategy or position in the market.

Finally, if the MACD increases hugely, compared to the past, this is because too many people have bought Bitcoin, and we are entering a 'bubble'. The bubble will burst, and the price should return to previous values. The opposite can happen, but is less common.

The last two features are more for the convenience of humans, to help them with intuition based strategies. As such, the first situation is easiest to programmatically incorporate, whereas the other two and more subjective.

An example of the MACD is below:

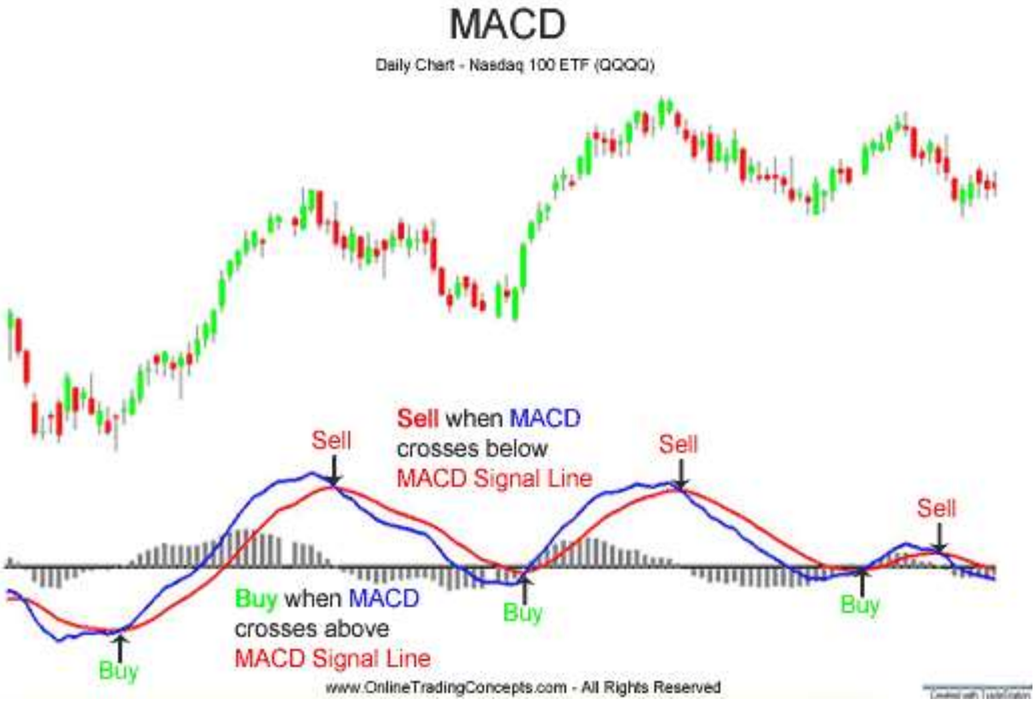


Figure 6. MACD graph. http://www.onlinetradingconcepts.com/images/technicalanalysis/MACDbuysellQQQQ.gif

INTUITION-BASED STRATEGIES

BUY AND HOLD

The strategy is extremely simple but effective in volatile markets. Buy at a seemingly reasonable price (compared to where the market is at), and then hold on to it for a long time, until there is profit. This strategy requires some sort of intuition and human decision making/risk taking, and thus to programmatically implement this, one would use a combination of an EMA, to determine the right time to buy in the current market, and a predefined time period to wait for.

This strategy is something to be aware of, as it is commonly used as strategy to benchmark against. It cannot really be implemented.

NEWS BASED

Bitcoin exchange prices are highly volatile and affected easily by outside factors, such as:

- Publicity – I observed several times that after the release of an article by the BBC or other large news provider, the price of Bitcoin was driven up. This is because the publicity causes people to enter the market out of interest. The demand becomes greater than the supply, and thus the price goes up.

- Service closures – The Bitcoin market has suffered several price hits.
    - One prime example is when the "Silk road" was seized by the FBI. The Silk Road was a drug selling and distribution site whereby buyers and sellers used Bitcoin to accommodate purchases. As soon as it was closed down, many buyers had no more use for Bitcoin, and thus began selling. The supply became greater than the demand, and the price was driven down by a sizeable amount (at least $50) within an hour.

    - The other example that comes to mind is the closure and bankruptcy of Bitcoin's largest exchange, Mt.Gox. Many traders lost confidence in Bitcoin and the market after it was announced that ~$4 million worth of Bitcoins were stolen from Mt.Gox, forcing the exchange to close. The general price of Bitcoin plummeted from around $1000/BTC by almost a half.

Therefore, a heuristics search on news items relating to Bitcoin is something worth noting, and was not investigated due to the complexity being outside the scope of this project. A proposed implementation could text the user about any news as soon as possible, whereby the user could text back an action. This approach would mean that the user would be able to react to the market fast enough to make a profit. Alternatively, a heuristics engine could analyse new articles and attempt to determine the mood of the article. Based on the mood, a certain trade action could be executed, however this is risky as heuristics based analysis is often not accurate.

IMPLEMENTATION

LANGUAGE AND TECHNOLOGIES

INITIAL CONSIDERATIONS

Initially, I considered a client-side implementation in the Delphi programming language. This means that the application would need to run on a local machine at all times. This did not seem like the right choice of technology, but at the time, I was not yet familiar with the paradigms required to build cloud applications. While it is a lot easier to design and implement an application in a desktop application style, it became apparent that this is less suitable than a web based application, which would run all the time remotely. I ended up choosing Node.js as a platform, which is programmed in JavaScript, as speed and scalability were important to me.

The database was required to store market data, and bot information. The choice was between:

- MySQL – A common and relatively lightweight relational database
- PostgreSQL – A heavier but extremely powerful relational database
- MongoDB – A new technology based on NoSQL databases (no relationships)

The key factor was performance. PostgreSQL initially appeared to be the worst choice, and MongoDB the best choice. After reading an article[14] explaining MongoDB, I was informed that I would not be able add features such as linking bots to user profiles. This was because MongoDB is a non-relational database, when there could be future relations. A performance comparison between PostgreSQL and MySQL[15] shows that MySQL outperforms PostgreSQL in simple tasks, but PostgreSQL is consistently faster in queries that are more complex. Therefore, I chose PostgreSQL.

NODE.JS JAVASCRIPT PLATFORM - BACKEND

Using JavaScript as a programming language seemed suitable for the task, since it was asynchronous. This means that the trader would pause while waiting for data from a market, instead it can carry on processing other tasks, until the remote data has been returned. Other advantages of JavaScript include fast debugging, an active support community, and plenty of libraries that can simplify some of the tasks.

Node.js is a relatively new platform, designed for running JavaScript on a server (traditionally, JavaScript runs in the web browser, for manipulating elements). It has already a wide range of 'modules' that can be painlessly installed, in order to speed up development. For example, the technical implementation of providing the BTC-e exchange with data in the correct format was already available in the form of a module. Node.js is extremely scalable, since there is no exponential increase as more users connect to the server, compared with PHP or Ruby.

Choosing Node.js was a risk, since I had never used it before and had minimal JavaScript experience. A single Node.js instance both acted as a web server, as well as carried out any trading. Node.js' asynchronous nature makes it a lightweight task to serve data to the user while also analyzing price data.

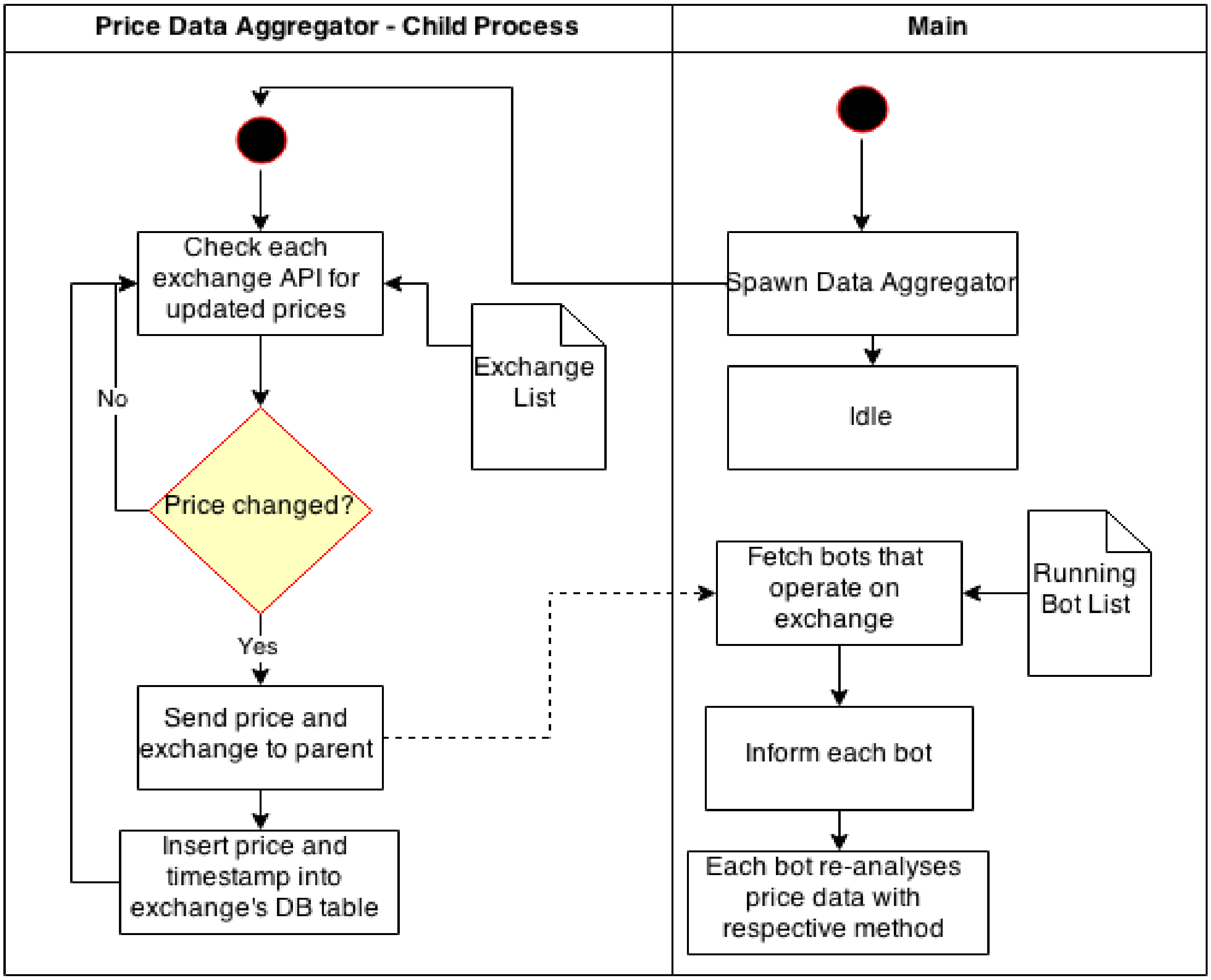
[14] Why you should never use MongoDB – See bibliography
[15] Comparing PostgreSQL 9.1 vs MySQL 5.6 using Drupal 7.x – See bibliography

Briefly, the tasks of the Node server were:

- Create multiple trading bots, that could run simultaneously with different parameters
- Download price data updates constantly and store in the database
- Inform bots when the price of their respective exchange has changed
- Request transformed OHLC data from the database via an SQL query
- Carry out the technical analysis on price data for each bot
- Handle buying and selling, inclusive of trading fees online
- Provide a simulation mode, whereby real prices are received, but fake trades occur
- Serve data accessible via a web browser

Because database operations are far heavier than the rest of the process, a child instance[16] was created by the server. This stops any database related items clogging up any other bots from trading when their data is ready. It is this instance's duty to check for new prices and insert them into the database. Management of the child is done through the "data aggregator" object.

A simplified flowchart of how the two process interact with each other is below:


Price Data Aggregator - Child Process
Main
Check each exchange API for updated prices
Exchange List
Spawn Data Aggregator
Idle
No
Price changed?
Yes
Fetch bots that operate on exchange
Running Bot List
Send price and exchange to parent
Inform each bot
Insert price and timestamp into exchange's DB table
Each bot re-analyses price data with respective method

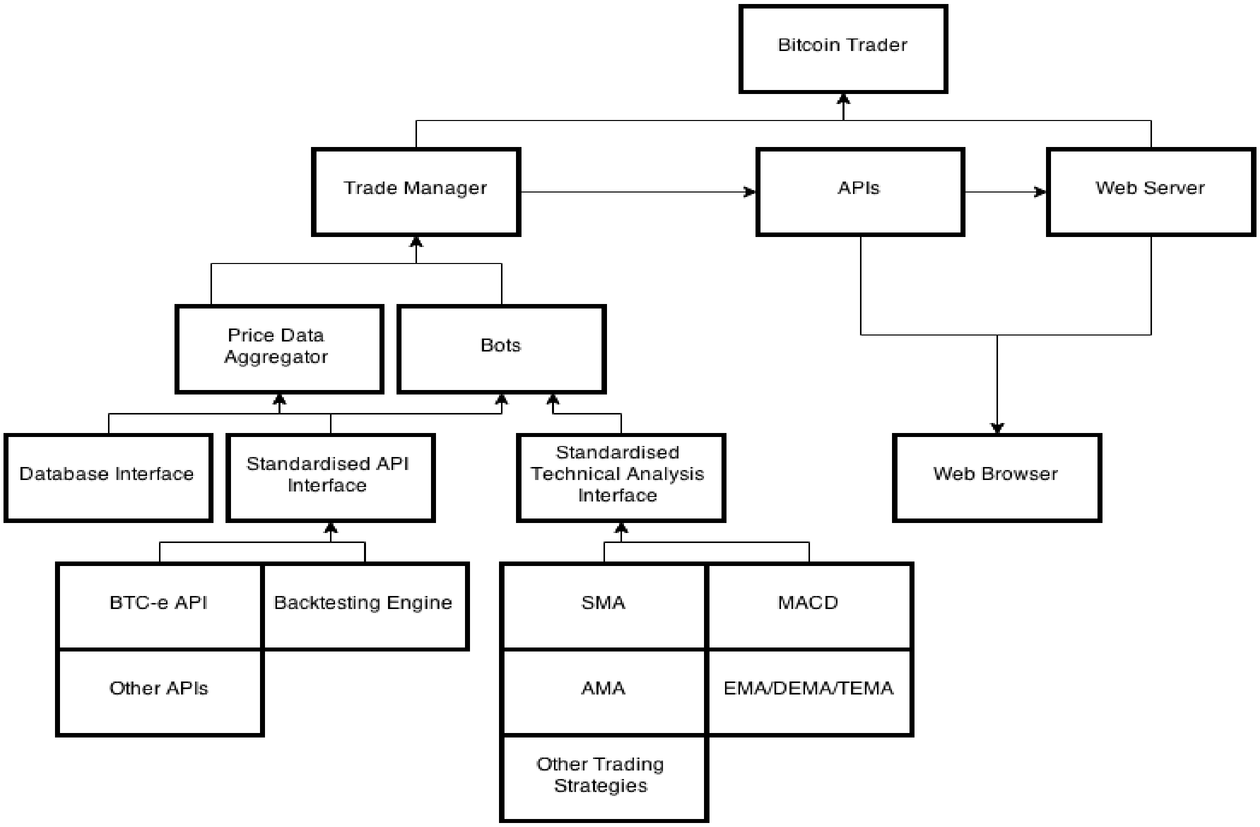
[16] See appendix 1.23

Thus far, the implementation has been from the point of a bot manager. We will now look at the implantation from a bot/worker perspective:

1) A bot is created, and provided certain parameters such as:
    - Market to trade on
    - Trading strategy and relevant parameters
    - Time interval of price data
    - If it is in simulation mode, and the parameters
2) The bot then creates an instance of the strategy advisor module, which understands what to do with the strategies and parameters.
3) If it is informed of a change in price data it will inform the strategy advisor module to:
    - Request the latest data in OHLC format from the database
    - Analyse the data with the given trading methods
    - Inform the bot if it should buy or sell Bitcoins
4) If the bot is informed to buy Bitcoins, and it is currently not holding Bitcoins, it will buy them at the latest price. Conversely, if it is a sell order, and it is holding Bitcoins, then it will sell at the latest price. Both conditions must be satisfied in either case for a successful order.

The trader itself was designed with a view to extensibility. Thus, the bots access all methods through a standardised interface. This is means that, for example, the underlying code behind accessing a certain exchange's price data can be changed, without having to modify anything else. This also means new exchanges can be "plugged in", even if the underlying code is completely different. This is also the case with the trading methods, allowing for new technical analysis indicators to be inserted whenever required.

A simplified structure diagram below (specifically the left) represents this:



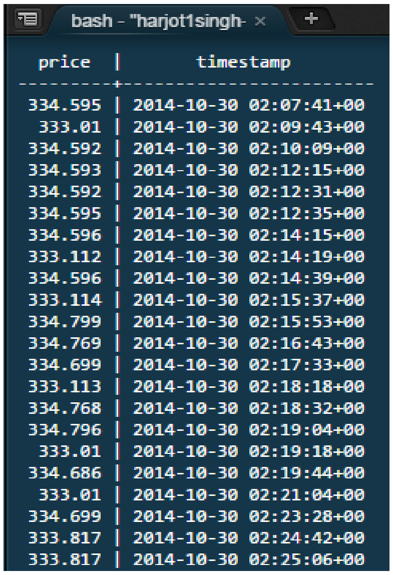
POSTGRESQL RELATIONAL DATABASE – DATA STORAGE

To acquire data, each Bitcoin exchange, at the minimum, presents an API[17] (application programming interface) that provides the latest trade data. By constantly requesting information from it, at a set interval of 1 second, we can catch any changes in the last trade price. The trade price and time of occurrence is inserted into the database.

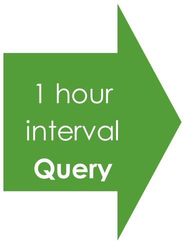
In terms of transforming and receiving data, PostgreSQL was a choice I was not certain about in the beginning. It seemed too heavy and unnecessary for simply storing and retrieving data. As I went along with it, I discovered it is far more efficient to allow the PostgreSQL server to do the data aggregation, and transform a list of timestamps[18] and prices into a list of times of set intervals, and the respective open, high, low, and close values for each time period. This was in contrast to sending all the data to the trader, and allowing the trader to do the aggregation, which means that far more data would have been sent, and processing power would be pointlessly consumed by the trader.

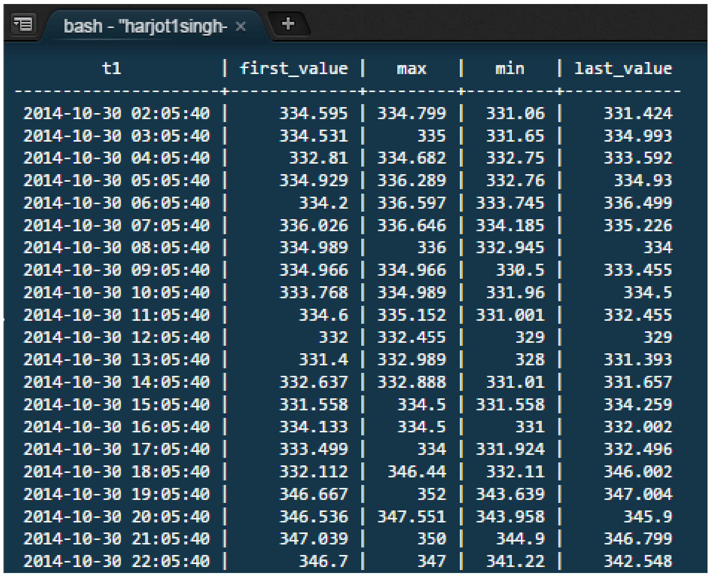
Therefore, I wrote a query[19] that, provided with a time interval, would convert the list of prices and timestamps into a list of times of a set interval and their respective OHLC values.

An example of the table structure and this transformation is below, using the SQL query:



```
  price   |       timestamp
----------+------------------------
 334.595 | 2014-10-30 02:07:41+00
  333.01 | 2014-10-30 02:09:43+00
 334.592 | 2014-10-30 02:10:09+00
 334.593 | 2014-10-30 02:12:15+00
 334.592 | 2014-10-30 02:12:31+00
 334.595 | 2014-10-30 02:12:35+00
 334.596 | 2014-10-30 02:14:15+00
 333.112 | 2014-10-30 02:14:19+00
 334.596 | 2014-10-30 02:14:39+00
 333.114 | 2014-10-30 02:15:37+00
 334.799 | 2014-10-30 02:15:53+00
 334.769 | 2014-10-30 02:16:43+00
 334.699 | 2014-10-30 02:17:33+00
 333.113 | 2014-10-30 02:18:18+00
 334.768 | 2014-10-30 02:18:32+00
 334.796 | 2014-10-30 02:19:04+00
  333.01 | 2014-10-30 02:19:18+00
 334.686 | 2014-10-30 02:19:44+00
  333.01 | 2014-10-30 02:21:04+00
 334.699 | 2014-10-30 02:23:28+00
 333.817 | 2014-10-30 02:24:42+00
 333.817 | 2014-10-30 02:25:06+00
```





```
         t1          | first_value |   max   |   min   | last_value
---------------------+-------------+---------+---------+------------
 2014-10-30 02:05:40 |     334.595 | 334.799 |  331.06 |    331.424
 2014-10-30 03:05:40 |     334.531 |     335 |  331.65 |    334.993
 2014-10-30 04:05:40 |      332.81 | 334.682 |  332.75 |    333.592
 2014-10-30 05:05:40 |     334.929 | 336.289 |  332.76 |     334.93
 2014-10-30 06:05:40 |       334.2 | 336.597 | 333.745 |    336.499
 2014-10-30 07:05:40 |     336.026 | 336.646 | 334.185 |    335.226
 2014-10-30 08:05:40 |     334.989 |     336 | 332.945 |        334
 2014-10-30 09:05:40 |     334.966 | 334.966 |   330.5 |    333.455
 2014-10-30 10:05:40 |     333.768 | 334.989 |  331.96 |      334.5
 2014-10-30 11:05:40 |       334.6 | 335.152 | 331.001 |    332.455
 2014-10-30 12:05:40 |         332 | 332.455 |     329 |        329
 2014-10-30 13:05:40 |       331.4 | 332.989 |     328 |    331.393
 2014-10-30 14:05:40 |     332.637 | 332.888 |  331.01 |    331.657
 2014-10-30 15:05:40 |     331.558 |   334.5 | 331.558 |    334.259
 2014-10-30 16:05:40 |     334.133 |   334.5 |     331 |    332.002
 2014-10-30 17:05:40 |     333.499 |     334 | 331.924 |    332.496
 2014-10-30 18:05:40 |     332.112 |  346.44 |  332.11 |    346.002
 2014-10-30 19:05:40 |     346.667 |     352 | 343.639 |    347.004
 2014-10-30 20:05:40 |     346.536 | 347.551 | 343.958 |      345.9
 2014-10-30 21:05:40 |     347.039 |     350 |   344.9 |    346.799
 2014-10-30 22:05:40 |       346.7 |     347 |  341.22 |    342.548
```

The basic steps that the query follow are:

1) Calculate a series of times of the provided time interval apart. E.g. 1 hour interval would produce a list of times of 00:00, 01:00, 02:00 and so forth.
2) Find out the trades that happened in each of the time intervals. E.g. Between 00:00 and 01:00, there were the following prices: 270.2 @ 00:20, 280.4 @ 00:32, 290.1 @ 00:57
3) Find out which of these were the highest, lowest, earliest, and latest, and set them to the OHLC values. In the previous example, this would be 270.2 (O), 290.1 (H), 270.2 (L), 290. (C)

[17] See appendix 1.18
[18] See appendix 1.19
[19] See appendix 1.24 and 1.25

ANGULAR.JS & TWITTER BOOTSTRAP FRAMEWORKS – FRONTEND

The frontend interface was produced in JavaScript, using a framework called Angular.js. When the user wishes to access a dashboard, containing the settings and performance of the user's bots, the user accesses the web link. The general style of buttons and interface was provided by the Boostrap framework, commonly used to get modern looking web applications running quickly. The browser downloads the initial webpage and template. Any items clicked on only requests the data required from the web server, as opposed to retransmitting the entire page with the navigation bar and headers. This approach also allowed for live graphs of price data and trends to be loaded and updated. The frontend was created primarily for presentation purposes.

CONCLUSION

In the end, it was possible to build an automated Bitcoin trader, based on the well-known technical indicators available. Whether it will be profitable, and how, is something to further research.

Some human interaction is required to set up the parameters on the trading methods, such as long and short EMA duration, and this is really down to user experience and preference. It is particularly worth noting that the most popular combinations, such as long EMA 21 against short EMA 11 were being used already on exchanges by a sizable amount of traders, and thus did not result in profit.

High frequency trading (more than one trade a second) was found to not be possible, purely due to the technical limitations the exchanges apply when accessing their APIs. A high frequency approach is unheard of in Bitcoin markets, and if it becomes possible, may allow for completely different and original strategies.

Bitcoin was the only crypto-currency of interest in this project. It is worth noting that many smaller crypto-currencies, such as Litecoin do exist. Litecoin, the second largest crypto-currency to Bitcoin, is traded for smaller amounts. Changes in the Litecoin and other market prices directly influence the Bitcoin market price.

In terms of project management, I should be more wary in the future of potential setbacks, and should plan to encounter them, even for seemingly trivial tasks. Research is best done by experimenting and reading about experiences backed by statistics. The technology chosen was correct, however I suspect that more time spent structuring the code initially would have led to less refactoring and cleaning up near the end of the project.

On the technical front, I have learnt how to make a web application using unfamiliar client frameworks, such as Angular.js, or the server-side JavaScript platform Node.js. Prior to this, my JavaScript ability was limited, and I have been able to get to grips with some of the more powerful features of the language, such as asynchronous callbacks.

Based on this, I believe that the following improvements could be made in the future:

EXPLORE MORE EXOTIC AND LESS TRADITIONAL STRATEGIES, SUCH AS THE "MARKET MAKER" ALGORITHM DESCRIBED IN THIS TWO-PART SERIES[20] - PROACTIVE TRADING

This project used reactive trading methods, such as moving averages. Moving averages are sub-ideal in the generally volatile Bitcoin market. This reactive trading occurs based on what has happened in the past. Perhaps a proactive approach may have been better, by analyzing the order books, and predicting the direction of the market. This could give the trader a unique advantage if the strategy is successful.

ABILITY TO COMBINE/MODIFY INDICATORS AND METHODS

Being able to restrict the action of indicators, such as the EMA, based on other indicators, may help reduce false signals drastically. As well as this, modifying the AMA as discussed previously, to use DEMAs or TEMAs may produce a faster reacting AMA. Using DEMAs or TEMAs in the MACD may also produce faster reacting result.

AUTOMATIC STRATEGY SELECTION

The trader could run a set of strategies simultaneously, and constantly monitor the profit levels for a certain time period. The bots should then use the most profitable strategy for the time period, until another strategy becomes more profitable. This has a huge risk factor, as switching strategies based on profit results from too long or short a time period could render the profit result useless.

IMPLEMENT A NEWS-HEURISTICS ENGINE THAT WOULD NOTIFY THE USER ABOUT EVENTS INSTANTLY, AND PRESENT DEFAULT ACTIONS (VIA SMS OR INTERFACE)

By scanning google news and common news websites, the trader could identify when Bitcoin has been mentioned. In the past, the publicity has meant that the price of Bitcoin goes up, as curious readers enter the Bitcoin market. Sentiment analysis on the article may be useful, as a strongly positive sentiment could result in a higher price change than a negative sentiment.

RESEARCH THE POSSIBILITY OF ARBITRAGE – BUYING/SELLING FROM ONE EXCHANGE, ONLY TO TAKE ADVANTAGE OF THE PRICE DIFFERENCES, AND SELLING/BUYING ON ANOTHER.

Arbitrage is a technique that involves exploiting the price difference between different exchanges. The exchanges usually set high trading fees, and impose a 24 hour period to process withdrawals. If there are any markets that offer instant withdrawal, arbitrage could become a serious option when automated.

TRADE BETWEEN DIFFERENT CRYPTO-CURRENCIES

This EPQ has only investigated Bitcoin. There are many other crypto-currencies out there, the next most prominent being Litecoin. As other crypto-currencies increase in value and volume, it could be wise to monitor these markets for similar arbitrage-like opportunities.

[20] Algorithmic trading with Bitcoin – part 1 – See bibliography

APPENDIX

1.1. **Peer-to-Peer (P2P)** – Direct communications between users of a network, as opposed to a centralised server or trusted party handling communications

1.2. **Crypto-Currency** – A digital currency whereby the creation and regulation is by cryptographic means, and thus is usually decentralized

1.3. **Cryptography** – A branch of mathematics that lets us create mathematical proofs that provide high levels of security

1.4. **Exchange** – A site that allows buyers and sellers to trade (Bitcoins)

1.5. **Liquidity** – A high volume of activity in a market

1.6. **Open** – The opening trade price in a given time interval

1.7. **Close** – The closing trade price in a given time interval

1.8. **High** – The highest price in a given time interval

1.9. **Low** – The lowest price in a given time interval

1.10. **Candlestick** – A format for storing the Open-High-Low-Close (OHLC) values for a given interval

1.11. **Bid** – The price a buyer is willing to pay (for a Bitcoin)

1.12. **Ask** – The price a seller is willing to accept (for a Bitcoin)

1.13. **Lag** – A late reaction to a market trend change

1.14. **Bullish** – When the majority of traders in a market are buying

1.15. **Bearish** – When the majority of traders in a market are selling

1.16. **Indicator** – A trading strategy that can indicate buy or sell signals

1.17. **Noise** – Price fluctuation in the market that confuses the interpretation of the trend

1.18. **API** – Application programming interface

1.19. **Timestamp** – The number of milliseconds since the epoch (January 1, 1970)

1.20. **Cloud Application** – An application/program that resides on a remote server, accessible via the internet. Data and all calculations are remote

1.21. **Paradigm** – A style of programming or design pattern

1.22. **Parent Instance** – The main application/program

1.23. **Child Instance** – A sub-process that is created by the parent instance. The parent instance has control over the child instance, however the child instance is also a completely separate process

1.24. **SQL** – Server Query Language. Used to request information from a database, by writing a query

1.25. **Query** – A SQL statement that returns data from a database in a certain form

1.26. **Proactive Trading** – A style of trading that predicts trends based on the prices people are willing to buy and sell for

1.27. **Reactive Trading** – A style of trading that predicts trends based on historical price data

BIBLIOGRAPHY

Adaptive Moving Average [Online] / auth. Forex Karma. - http://www.forexkarma.com/adaptive-moving-average.html. - Explains everything about Kaufmann's adaptive moving average, from how to calculate it, to using it as a trading indicator.

Algorithmic trading with bitcoin – part 1 [Online] / auth. Firth Paul // Wildbunny Blog. - http://www.wildbunny.co.uk/blog/2014/06/11/algorithmic-trading-with-bitcoin-part-1/. - 1 part of a series of articles that describe potential "proactive trading" strategies.

AngularJS Controller Tutorial With Example [Online] / auth. Patel Viral // Viral Patel's Blog. - http://viralpatel.net/blogs/angularjs-controller-tutorial/. - An in-depth tutorial demonstrating the use of Angular.js controllers, used for manipulating the frontend.

AngularJS Step-by-Step: Services [Online] / auth. Cooper Jim // Pluralsight Blog. - http://blog.pluralsight.com/angularjs-step-by-step-services. - A tutorial explaining the usage of Angular services, to create reusable code.

Bitcoin: An Electronic Peer-to-Peer Cash System [Online] / auth. Nakamoto Satoshi. - https://bitcoin.org/bitcoin.pdf. - The original paper published describing the abstract and possible implementation of the Bitcoin protocol. Satoshi Nakamoto disappeared a short while after Bitcoin was implemented, and does not appear to be a real person.

Bootstrap Tutorial [Online] / auth. Tutorial Republic // Tutorial Republic. - http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/. - A site containing how to do everything with Bootstrap for styling the frontend.

Calculating Exponential Moving Average - a Tutorial [Online] / auth. iexplain // iexplain.org. - http://www.iexplain.org/ema-how-to-calculate/. - How to programmatically calculate the EMA.

Candlesticks Explained [Online] / auth. Knownware. - http://knownware.com/wp-content/uploads/2012/11/Candlestick-Example.png. - An image depicting the 2 types of candlesticks.

Comparing PostgreSQL 9.1 vs. MySQL 5.6 using Drupal 7.x [Online] / auth. O'Sullivan Padraig. - http://posulliv.github.io/2012/06/29/mysql-postgres-bench/. - A statistical comparison between MySQL and PostgreSQL. Shows that PostgreSQL generally performs much better..

CREATING A SIMPLE RESTFUL WEB APP WITH NODE.JS, EXPRESS, AND MONGODB [Online] / auth. Buecheler Cristopher // CWBUECHELER. - http://cwbuecheler.com/web/tutorials//restful-web-app-node-express-mongodb/. - A thorough guide explaining how to create an web server using node.js, with an example database.

Double (D-EMA) and Triple Exponential Moving Average (T-EMA) [Online] / auth. Brown Derry // ETF HQ. - http://etfhq.com/blog/2010/11/17/double-and-triple-exponential-moving-average/. - Compares the EMA, DEMA and TEMA, and explains how they can be calculated.

End of the Silk Road: FBI Busts the Web's Biggest Anonymous Drug Black Market [Online] / auth. Forbes // Forbes. - http://www.forbes.com/sites/andygreenberg/2013/10/02/end-of-the-silk-road-fbi-busts-the-webs-biggest-anonymous-drug-black-market/. - Forbes, discusses the Silk road shutdown and how the arrest of the owner took place. The Bitcoin price was driven low after this, as users were selling their Bitcoins.

Exponential Moving Average (EMA) – Technical Indicator [Online] / auth. Hill Alton // TradingSim Day Trading Blog. - http://tradingsim.com/blog/exponential-moving-average-ema-technical-indicator/. - Explains how to calculate the EMA, and has some example graphs of the EMA being applied to price data.

Exponential Moving Average (EMA) Graph [Online] // Online Trading Concepts. - http://www.onlinetradingconcepts.com/images/technicalanalysis/MAexponentialEBAY.gif. - A graph that helps show the difference between the EMA and the SMA.

Hourly Candlestick Chart [Online] / auth. Incredible Charts. - http://static.incrediblecharts.com/images/2011/2011-12-24-gold.png. - An example image of a candlestick graph with a time interval of 1 hour.

Jade vs HTML: Why it matters [Online] / auth. Stark Vincent // Stark's Blog. - http://vstark.net/2014/10/18/jade-vs-html/. - An article and demo that convinced me to use Jade instead of HTML, to write cleaner code.

jQuery vs. AngularJS: A Comparison and Migration Walkthrough [Online] / auth. Lamb Daniel // Airpair. - https://www.airpair.com/angularjs/posts/jquery-angularjs-comparison-migration-walkthrough. - A comparison of the jQuery and Angular.js frontend frameworks.

Kaufman's Adaptive Moving Average (KAMA) [Online] // FxCodeBadeWiki. - http://fxcodebase.com/wiki/images/thumb/d/d7/KAMA1.png/640px-KAMA1.png. - A graph showing the KAMA vs the AMA.

MACD Moving Average Crossovers [Online] / auth. Online Trading Concepts // Online Trading Concepts. - http://www.onlinetradingconcepts.com/images/technicalanalysis/MACDbuysellQQQQ.gif. - An image that shows the possible situations and actions in a MACD.

Node-Postgres [Online] / auth. Carlson Brian // Github. - https://github.com/brianc/node-postgres. - A node module and tutorial for using the postgres database.

Users Track $100 Million in Stolen Bitcoin After Sheep Marketplace Hack [Online] / auth. Bradbury Danny // CoinDesk. - http://www.coindesk.com/sheep-marketplace-track-stolen-bitcoins/. - Around $100 Million worth of Bitcoins stolen from drug site caused crash in markets.

Where Did the Bitcoins Go? The Mt. Gox Shutdown, Explained [Online] / auth. Brustein Joshua // Bloomberg Business. - http://www.bloomberg.com/bw/articles/2014-02-26/where-did-the-bitcoins-go-the-mt-dot-gox-shutdown-explained. - A Bloomberg article giving some insight into the shutdown of Mt.Gox, the largest Bitcoin exchange.

Why database connection object or connection pool needs to be made singleton? [Online] / auth. Authors Multiple // Stack Overflow. - http://stackoverflow.com/questions/13521660/why-database-connection-object-or-connection-pool-needs-to-be-made-singleton. - A stack overflow Q&A about the point of connection pooling.

Why you should never use MongoDB [Online] / auth. Mei Sarah // Sarah Mei's Blog. - http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/. - Although it seems like a biased source, the article fairly evaluates the different database technologies, based on personal experience and public statistics/facts. It explains clearly why MongoDB is a bad choice for a project that could potentially extend.