Comenius University in Bratislava
Faculty of mathematics, physics and informatics

Comovement between cryptocurrencies

Bachelor thesis

2019
Michal Porubský

Comenius University in Bratislava
Faculty of mathematics, physics and informatics

Comovement between cryptocurrencies

Bachelor thesis

Study programme:	Computer science
Field of study:	Computer science
Training work place:	Department of computer science
Supervisor:	RNDr. Ing. Peter Molnár PhD.

Bratislava, 2019
Michal Porubský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Porubský
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Comovement between cryptocurrencies
Podobný pohyb kryptomien

Anotácia: Skúmať podobný pohyb kryptomien, odhadnúť možné dôvody a využiť túto informáciu v implementácii obchodnej stratégie.

Vedúci: Ing. RNDr. Peter Molnár, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 08.11.2018

Dátum schválenia: 08.11.2018 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Michal Porubský
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Comovement between cryptocurrencies

Annotation: Searching for patterns of comovement between cryptocurrencies, assessing possible reasons behind it and implementing trading strategy based on this comovement.

Supervisor:	Ing. RNDr. Peter Molnár, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.

Assigned: 08.11.2018

Approved: 08.11.2018 doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Abstrakt

V tomto texte sa zaoberáme štúdiom podobného pohybu medzi kryptomenami. Krypto trh je zaujímavý tým, že je nový, nepredvídateľný a neprebádaný. V úvode sa pozeráme na kryptomeny ako také a vysvetľujeme si špecifiká ich trhu. Ďalej popisujeme dostupné zdroje dát - cien kryptomien a opisujeme implementáciu aplikácie, ktorá stiahne všetky ceny v čase tých kryptomien, ktoré spĺňajú naše požiadavky. Bližšie sa pozeráme na analýzu dvojročného cenového vývinu tridsiatich deviatich kryptomien a pomocou euklidovskej vzdialenosti a kointegrácie určujeme dvojice kryptomien, ktoré preukázali spoločný pohyb. Využívajúc spomínané informácie navrhneme a implementujeme obchodnú stratégiu neutrálnu voči vývoju trhu a predostrieme výsledky, ktoré naša stratégia vynesie. Pozrieme sa aj na obmeny obchodnej stratégie a ich výsledky. Na záver si ukážeme, ako sa naša obchodná stratégia správa na pätnásť minútových a jednodňových dátach.

Kľúčové slová: spoločný pohyb, kryptomeny, párové obchodovanie, vysokofrekvenčné dáta

Abstract

In this text we study comovement between cryptocurrencies, a new and unusually volatile market. We start by discussing cryptocurrencies and the nature of its market. Then we describe the available sources of price data and implement an application that fetches all the available data relevant to our needs. We look closer at two years of price data of one hour frequency for thirty-nine cryptocurrencies and use euclidean distance and cointegration algorithms for analyzing comovement between the cryptocurrencies. Using the information about comovement pairs, we implement a market neutral trading algorithm and discuss the out-of-sample results considering modifications of the algorithm. Lastly, we discuss how our trading algorithm performs on data of fifteen minute and one day frequencies.

Keywords: comovement, cryptocurrencies, cointegration, pairs trading, high frequency

Contents

List of Figures

List of Tables

Introduction

The aim of this study is to study comovement between cryptocurrencies, analyze it and implement a trading strategy based on the findings. Comovement relationship between the assets of various markets is well studied. Let us mention comovement in the commodity market [29], brent crude futures market [23] and international stock markets [27].

There is not enough research considering the crypto market, though. In [25], the author discussed the volatility comovement of Bitcoin relative to Ether, the two major cryptocurrencies. However, a more generic approach considering bigger number of cryptocurrencies with a practical implication is still missing.

Since cryptocurrencies are becoming increasingly popular and this market is unusual and new, studying comovement in this market is an interesting field for research. Therefore we chose to look deeper into it and implement a trading strategy that uses the information.

We start by introducing cryptocurrencies in chapter 1. We discuss what cryptocurrencies actually are, what their purpose is, why they are so important for the future world, what determines their price and why their market is so volatile.

In chapter 2 we define what we understand by comovement of cryptocurrencies, show that it's not just similar movement, but rather any relationship between the price development. We then review some of the well known statistical methods for studying comovement.

In the next chapter, 3, we look into details on what data we base our study on. We review the factors of choosing the right data source, remove a slight market trend and choose a source of data for data used in this text.

In the following chapter 4, we introduce our solution for fetching the data, given certain constants such as price frequency and minimal wanted data length. The actual implementation is included on the DVD included with this thesis. We discuss requirements we expect from the application, the stack we chose to help with that, architecture and exact implementation. We also describe problems we faced together with their solutions.

In chapter 5 we describe stack we used to analyse the fetched data and look deep into it. We preprocess the data, explain how and why and discuss the actual implementation of comovement tests suggested in chapter 2. We also include many examples and graphs. Runnable code of the analysis described is located on the DVD included with this text.

Finally in chapter 6 we go step by step on how we built our trading algorithm that uses the comovement relationship of cryptocurrency prices. We discuss possible approaches, explain what advantages they include and choose a trading strategy that is market neutral and bets only on the actual comovement relationship and nothing else. We also explain what parameters are customizable and prepare the ground for the finale chapter.

In chapter 7, we reveal our findings and results we had experimenting with different input parameters into the trading algorithm, with different data frequency and more. We also try to guess possible reasons behind such results.

Chapter 1

Cryptocurrencies

In this chapter we cover basic information about cryptocurrencies and the specifics that make them interesting to analyze.

1.1 Definition

According to [21]:

> A cryptocurrency is a digital or virtual currency that uses cryptography for security. A cryptocurrency is difficult to counterfeit because of this security feature. Many cryptocurrencies are decentralized systems based on blockchain technology, a distributed ledger enforced by a disparate network of computers. A defining feature of a cryptocurrency, and arguably its biggest allure, is its organic nature; it is not issued by any central authority, rendering it theoretically immune to government interference or manipulation.

1.2 Purpose

There is a number of cryptocurrencies in the world (2104 as of the 2019/02/04 [4]), each serving a different primary purpose and hence running on different technology and using different schema for security, validation of transactions, **mining** of the cryptocurrency, and other. Mining refers to the ability of exchanging computational power for a fraction of the asset. Mining is usually crucial for the cryptocurrency network to work. For a brief overview we offer following categorization of cryptocurrencies based on their inner purpose:

- Alternative currency, with focus on:
  - Digital money

    * **Bitcoin** (BTC) - The first cryptocurrency ever that introduces a distributed money system the other cryptoassets are building on top of. Mining is done by the computation of a problem that is computationally hard, even though the solution to the problem itself is not of any value. The problem is adjusted in such a way that the speed of it being solved is regulated and thus predetermines the increase of the flowing cryptocurrencies in the system over time. It relies on some of the well known mathematically hard problems.

  - Privacy
    * **Monero** (XMR) - In most cases, cryptocurrencies have transparent transactions with the only problem identifying the real-world person being *behind* the account. Monero uses cryptography that enables obfuscating both addresses (accounts' identification) and the amount of money included in the transaction, thus it proves as a way more private way for sending money [22].
    * **ZCash** (ZCH) - Similar to Monero in terms of purpose, with difference being the ability to choose what to share with others, hence theoretically making the system a little less private as it is possible for an authority to **force** you into revealing your secrets.
  - International (thanks to the distributed nature) **fast** transactions
    * **Ripple** (XRP) - Blockchain technology used as a means of banks sending money fast to each other. It doesn't offer to make your own wallet.
    * **Stellar** (XLM) - Super fast sending of money across any two users. It was once based on a system similar to Ripple, but diverged some time ago and came with a new one.

- Distributed computing
  - **Ethereum** (ETH) - A blockchain app platform that provides the computational power for applications developed on the network by making the mining process designed exactly for that - mining basically means solving a problem an app is willing to pay for. In other words, mining earns the miner *money* the app provides for solving a problem the app wants to be solved. This makes the mining process more meaningful in contrast to Bitcoin.

Since most of the coins are open source and there is usually no authority to regulate them, there are often moments in time when developers have different opinions about the future development and **fork coins** are created. Since they build on common existing codebase, it is quite a strong technical correlation and may prove in comovement of the prices as well. For an illustration we provide some examples:

- **Ethereum classic** (ETC) - Forked from Ethereum after the hacker attack stealing milions-worth of the currency from Ethereum's website for funding start-ups. Ethereum as we know it updated the code of its system in order to devalue the stolen asset. Ethereum classic argued that such an intervention from the system should not be possible and makes the currency not trusted. It was more of an ethical question that resulted in Ethereum classic fork that did not modify the code and kept accepting the stolen money and commonly-used Ethereum that modified the code.

- Bitcoin cash (BCH) - Forked from Bitcoin after the majority behind Bitcoin turned down the idea to scale the platform by modifying the size of transaction block (group of transactions to be verified and agreed upon by the network) resulting mostly in speed performance.

1.3 Volatility

1.3.1 The price of an asset

The most of cryptocurrencies are not backed by any money at all. It is just a system that gives the value to itself, starting from zero. There are some exceptions, though. For example Ethereum based cryptocurrencies buy some initial amount of Ethereum to run on and divide the value among the new coins. The price of a cryptoasset is relative. Cryptoasset is given its value by sentiment only. The official way to get some of the currency is to **mine** the currency.

However, there is an unofficial way of getting the asset called **trading** on exchanges. The idea is simple: Someone owns the asset and offers it for some money or a portion of a different asset per unit. You agree and you two exchange your possessions, usually for a trading fee taken by the exchange.

1.3.2 Implications

The fact that cryptocurrencies' price is determined by sentiment of the people trading imply that the price is highly volatile and unpredictable. It is not unusual for the prices to increase or drop even by 10% on a daily or hourly basis. Thanks to the high means of price deviations it proves attractive for speculative traders that aim to maximize profit by buying and selling the asset in the right time. Thus, means of predicting the price prove highly valuable. One of the aims of this study is to try different strategies that can be used to predict future development of prices of a group of related assets.

Chapter 2

Comovement

In this chapter we discuss comovement and go through some of the well known methods for analyzing it. We then pick some of them for later use in the study and justify our selection.

By comovement between two or more cryptocurrencies we understand any high enough correlated or by any means similar development of the relative trading prices of the currencies.

It is important to note that all the cryptocurrencies are correlated to some extent, given that the crypto market is new and that it is often used as a means of investing money. As a result, it falls victim to many panic buy and sells, so called pumpings and fluctuations of the price on a few minute basis. Pumpings represent sudden investment of money to a cryptocurrency with low volume. Even a little money can cause a sudden spike of 1000%, fooling many people and trading bots looking for the best investment pair and quickly sells the security back for a much higher price.

The knowledge of which cryptocurrencies form a firm comovement groups could provide a trading advantage. Let's illustrate that on an example:

> Let assets A, B, C form a firm comovement group. Imagine a sudden spike of the price of A. Since A, B and C are correlated, we believe that B and C follow the spike. So we buy more of B and C to sell for a higher price. Analogically, for a price drop, we sell B and C in order to re-buy for a smaller price.

2.1 Strategies

2.1.1 The Distance approach

For a pair of cryptocurrencies, the Euclidean distance approach computes the sum of squared Euclidean distances between the two corresponding normalized price series.

The Euclidean distance is the length of a straight line in the Euclidean space between the two points in time. By price series, we mean the series of price development over time, so the Euclidean distance would be equivalent to the difference of price for a given time point. The normalization of the prices is required in order for our analysis to be relative to the prices of the two assets and to not overlook perfectly correlated pairs whose prices are just far away from each other.

This method is a very simple one. It is easy to implement and advocated in many studies performed on futures trading [26]. However, the crypto market is different. It is not unusual for the crypto market to gain 10% and fall back in a matter of minutes. Therefore we suppose that using this approach would not prove very profitable. For its simplicity, however, we chose to implement this method as well in order to have something to compare the results to.

2.1.2 The Cointegration approach

A pair of cryptocurrencies is said to be cointegrated if all of its price series are integrated to order one and their linear combination is integrated to order zero and mostly noise. Since it is a well-known and studied method, we won't go into much details regarding the method itself in this text. Rather we encourage readers keen to know more about the statistical and mathematical background to look into [12], [26], [23] or [24].

The intuition behind is that the cryptocurrencies are cointegrated if their linear combination tends to move around a constant. Let $X_1, \cdots, X_n$ be price series of n distinct cryptocurrencies. Let $c_i \in \mathbb{R}, i \in \{0, \cdots, n-1\}$. If they are cointegrated, the following holds:

$$Y = \sum_{i=0}^{n-1} c_i X_i$$

Y is a new price series that is mostly noise. As such, it moves around its mean a lot. This property is known as **mean reverting** [24]. In comparison to distance approach, this method allows for a more complex price correlation and is not influenced by sudden exponential spikes. Hence we suppose it may prove far more profitable.

Chapter 3

Data

In this chapter we explain what data we need, go through the available sources of it, explain its nature and select one our study is based on.

3.1 Data we need

Our study is based on **price data only**. There are other factors that could possibly say something about the comovement relationship, however, that is out of scope of this study. We encourage other studies to take on and research more with the inclusion of other aspects, such as:

- Actual volume of trading of the currency.
- Time of active trading on the exchange.
- Means of measure of active development behind the cryptocurrency. CryptoMiso [5] measures that in terms of new github commits over time and shows the data in handy graphs.
- And many more..

3.2 Price data

In 1.3 we explained what determines the value of a cryptoasset and showed that it's the people that trade it. However, there is one complication:

Trading is by definition exchanging some of the asset A for some of the asset B. Let A be a cryptocurrency we want to get the price for. It is tradable to B. B can be either fiat currency (any government accepted standard currency, for example EUR or USD) or another cryptocurrency. We can obtain just the price of A **relative** to the price of B from exchanges. Let's dig deeper into the problematics:

3.2.1 Fiat tradable cryptocurrencies

This is probably the most convenient for people trading manually, the reason being the simplicity of thinking about the cryptocurrency's value being equal to some amount of money. There are not many currencies that are tradable to fiat, though.

3.2.2 Cryptocurrency tradable cryptocurrencies

This is much more common in the world of exchanges. Even if just one cryptoasset is tradable to fiat, on most exchanges there is a possibility to converge from any cryptocurrency by a finite number of trades to it. There is a trade fee for every trade, though.

There are two special cryptocurrencies in the trading world:

- **Tether** (USDT) - Formerly it claimed that every one Tether was backed by one American dolar. It is not that way now [28], but its price relative to USD still stays pretty much identical. More often than not, it's possible to encounter an exchange that does not support a fiat currency at all. It uses Tether for that.
- **Bitcoin** (BTC) - In 1.2 we mentioned that it is the oldest cryptocurrency. As such, it has the biggest trading volume and hence is the most stable among the currencies in the crypto market. As such, trading relative to Bitcoin gives us these advantages:
  - It is used as the major trading currency in most exchanges, yielding the biggest number of both pairs and data per pair we can get.
  - Looking at the prices relative to the price of Bitcoin removes the aforementioned similar movement between all of the crypto market (2).

Because of the mentioned benefits of choosing Bitcoin as the quote currency (trading assets to Bitcoin and vice versa), we chose to reference to the price of an asset for a given time as the price of the asset relative to Bitcoin at that time.

3.3 Source of data

There are 256 exchanges as of the 9th of May, 2019 [4] that provide trading.

There is a popular trading strategy called **arbitrage** that makes the price across different exchanges as well as the computed price between the same asset across different tradable pairs pretty much the same. It basically stands for a simultaneous, risk-free buying and selling of the same asset on different platforms or the same asset

in different forms (via different chains of tradable pairs) on the same platform that benefits from the imbalance of prices and corrects the prices by doing so.

However, different exchanges provide different user experience, fees for buying and selling, security of the assets and more, making them unequal in the terms of amount of people using the product. The less people that use an exchange, the lower the trading volume and the less the probability for a fair price, making the exchanges with high volume the best choice for our research. Please note, that if the volume is practically non existent, there is no chance for even arbitrage behaviour - there's no one to sell and no one to buy on such an exchange.

3.3.1 Amount of data

For our methods to perform best, we need the biggest amount of data we can possibly get our hands on. This is a big problem, since cryptocurrencies as such are relatively new to the market and hence they lack a big enough history picture for us to analyze and build our theory upon.

3.3.2 Frequency

Since the price of cryptocurrencies is so highly volatile and there is an increasing trend for the use of trading bots (a computer program that, based on any strategy, buys and sells assets automatically, without the need of interference from a human), the prices fluctuate on an hour, 30 minute or even a minute basis. We consider multiple possible frequencies, ranging from a minute-based frequency of price data up to one day-based frequency.

3.3.3 Available API

Many exchanges provide good enough API for both trading and getting the history prices from. It is used by trading bots and statistics mainly. In order for us to conveniently collect data to base the research upon, the available API quality is a valid criterion.

There are multiple commonly used and old enough exchanges that accomodate our needs. After thorough consideration and analytics of the points above, thanks to statistics done on [4], we chose **HitBTC** as the means of getting our price data from.

In the following text we refer to currency prices as their prices relative to Bitcoin. For a better demonstration, we do not refer to the currency itself, but rather to its

currency pair with Bitcoin. For example, instead of writing *XMR* for the Monero currency, we write *XMRBTC* for the currency pair.

Chapter 4

Data crawler

In this chapter we discuss the implementation details behind the crawler we implemented for getting the prices and normalizing them into a nice readable format suitable for subsequent research. The runnable code is located on the DVD included with this thesis. See instructions in the README.md file.

4.1 Requirements

We want to analyze price data. For being able to do that, firstly we need to get the data. The crawler part of the implementation should do exactly that by using the available HitBTC API [8]. It should fetch all history prices for all relevant cryptocurrencies on the HitBTC market, given:

- **Quote currency** - We chose BTC as the quote currency for our study in 3.2.2. However, we want to build easily modifiable crawler, so we want to generalize it.
- **Data frequency** - Frequency of the data to fetch. One of M1 (one minute), M3, M5, M15, M30, H1 (one hour), H4, D1 (a day), D7, 1M (a month).
- **Minimal data length** - Length of the price development in months we want to analyze for the cryptocurrencies. Fetch prices just for cryptocurrencies that allow for such a long development and leave out those that are on the market for a shorter period of time.

Afterwards it should convert the fetched data into a format easily readable by subsequent analysis program and save it.

4.2 Stack, architecture

For the purposes of fulfilling the requirements we did not need to implement a complex system, nor we needed to implement a frontend for displaying and manipulating

with the arguments. Hence we chose to write simple scripts that would import input arguments (quote currency, data frequency and minimal data length), call each other and together do the job.

For that, we found useful the use of Node.js [1]. Since Javascript is the main frontend language used across the world wide web, fetching requests and manipulating the data is native to the language. It has a nice library ecosystem [11] and together with ES6 syntax sugar [3] compiled by Babel [2] it proves as a nice, easy and readable language to use.

We chose the combination of eslint [6] with prettier [13] as linters to make the code readable and unified and git [7] as a version control system, to easily track changes, revert back if needed and allow for multiple features development *at once*.

4.3 Implementation

All the main logic - scripts are conveniently located in the fetchAPI object of the app. Constants, input parameters, request builders and other utils are separated. In the following text, **symbol** stands for a trading pair. Let's go over the scripts:

- **Fetch symbols**. There is a convenient HitBTC API for getting all tradable symbols (currency pairs). We fetch that and filter them based on the quote currency input parameter.
- **Fetch data length**. Every tradable currency was announced to the market at a different point in time. The aim of this script is to get for each symbol the number of months the pair is tradable. We do that by fetching all its prices with a month frequency and counting the length of the fetched data.
- **Get relevant symbols**. This script goes through the list of tradable cryptocurrencies along with its tradable length (the output of the previous script) and filters them in such a way that it returns just those that are tradable for strictly greater number of months than our minimal data length input variable. We use strictly greater for ensuring that we select just those which are present on the market for a **full** number of required months.
- **Fetch prices for a symbol**. Given a symbol, the purpose of this script is to fetch all the available prices for it. There is a HitBTC API endpoint for getting the prices, but the maximum limit of prices per request is 1000. Since there is much more data for most cryptocurrencies, we need to paginate the requests.

We know the time the symbol entered the market, so this script generates all the timestamps of the given input frequency parameter from the start till the end with the step of 1000 (the max limit of prices per request) and maps the timestamp to request. Consequently when it has all the data, it just joins it into a handy hash map, the key being the timestamp of the price and the value representing the price.

- **Transform prices to an array**. Given a symbol, this script generates an array of a common size of the minimum data length's input parameter mapped to the wanted frequency and leaves either the price of the symbol at the given timestamp or null. Nulls happen to occur on such timestamps when no trade happened for the symbol. In that case we do not get that price from the API - we have to handle that appropriately in the above script by mapping the prices not to indexes but rather to the timestamps (since we may get the same timestamp with price more than once - that way we would just overwrite the value).

- **Fetch all**. Fetches relevant symbols (the 3rd script) and for each it gets the uniformed common size array of its prices or of nulls by calling the transform prices' script.

4.4 Problems

During the implementation, many problems arose. Some of them:

- **Long running time**. There are many requests going on. As of the 9th May 2019, there are 888 tradable currency pairs on the market, out of that 331 are tradable to BTC. That's 1 request to get that info. Plus 331 requests to get the info about the available data length of the corresponding currencies, plus many many more for getting the actual prices for all the timestamps of the given frequency. Imagine getting a minute frequency data for a duration of 2 years of one currency. The app would need to make $\left\lceil \frac{2\times365\times24\times60}{1000} \right\rceil = 1052$ requests. That's just too many for it to run fast.

  Solution: Asynchronicity. When making a request, we do not need to block computing and wait for the response but rather we can continue and do computations *in parallel*. It is not a good phrase to use, since Javascript does not actually support running in multiple threads, but it is a good example of what I mean. The code is not blocked by waiting for the response and we can perform faster. We can even perform multiple requests *at once* and wait for them *together*.

- **Too much heap memory consumed**. Since all of the data is really much (the minute frequency data for 40 cryptoassets consumed over 900MB of memory), we want to have it in memory for as short as possible. Asynchronically getting of all the currencies may not be the best idea, since we have all the data in memory at once.

  Solution: Differentiate between synchronous and asynchronous computing and use asynchronicity only where it makes sense.

- **Too many requests**. The API has a security mechanism to prevent being too slow or even irresponsive by controlling how fast and how often we can request certain endpoint from the same IP address. Due to the above inclusion of a certain non trivial degree of asynchronicity into our system, we needed to build a mechanism to prevent the API from blocking our requests.

  Solution: By trial and error we found out that 10 simultaneous requests waiting for 100 miliseconds after each successful response is enough for the API to never block us. We implemented that behaviour by overriding the request function for fetching an endpoint to be controlled by a **semaphore** of a size of 10, waiting for 100 miliseconds right before releasing the lock.

- **Getting the data all over again**. The data is a history data. It does not change. However, during the development of either the crawler of subsequent analysis we were changing the mind a lot, trying all possible input variable combinations and so on. Since it's so many requests, we did not want to wait for the functions to get all the data all over again every time something was modified.

  Solution: Caching middleware. Each slow enough function can be wrapped by the caching middleware and each time the function runs, first the middleware checks whether we have the data already computed and saved and only if we do not have it does it run the function with given parameters and then saves the result. If we have it, it just reads the output from the file where it is saved and returns. The important thing here is the caching key. We decided that a function call is considered identical to the same function call if and only if the input parameters of the app are the same (minimal data length, quote currency and data frequency) and the function is called with the same parameters. Naturally it is also differentiated based on the function called. Finally, the cache is not enough to be present in-memory, we need to actually store it on the hard storage, since the scripts does not handle errors and do not provide any user interface. They

just attempt to fetch everything and exit. In-memory gets wiped between the runs.

Chapter 5

Analysis

In this chapter we describe stack we used to analyse data. We plot the data to visually inspect it, discover some crypto specifics and prepare the ground for the upcoming trading algorithm. Runnable code is once again located on the DVD included with this thesis. See instructions in the README.md file.

5.1 Requirements

Once we got the data (see 4), we want to analyze it and make use of some of the comovement statistical strategies (see 2) in order to implement a successful trading algorithm. To do that effectively and to minimize our effort, we want to choose tools designed exactly for that. Our needs include:

- **Data transformation** Our data needs preprocessing for the methods to work. For example, there are gaps (4.3) in the data when no trade happened. Even the trading as such needs a lot of data transformation and computation. The point is, we are doing a lot of that. We need a tool that allows for as easy data transforming as possible, without compromising performance (we have a **lot** of data!).

- **Statistical methods** Implementing well-known statistical methods is not the aim of this thesis. It has been done already. With a bit luck, we would make use of existing libraries to make care of the statistics.

- **Plotting and describing** Aside from relying purely on our logic and having absolute faith in the algorithms we produce, we see a crucial role in plotting the data and trades into graphs. It showed to be of a great value. It enables us to actually visually see what's happening inside. A nice and simple way to plot data into multiple different graphs depending on the occasion is crucial for data analysis. As John Tukey mentioned in his famous book about data analysis [30]:

> The greatest value of a picture is when it forces us to notice what we never expected to see.

5.1.1 Stack

We found the use of Python [20] especially useful for meeting the requirements. It provides us with some very powerful and useful libraries for data manipulation, processing and statistics, making it one of the best choices for data science as such. To highlight just a few of them:

Pandas [15] provides very useful data structures similar to those of SQL tables called *DataFrames*. It is possible to store all the prices of all our assets into one DataFrame and work with that. It also provides data transformation, statistical description of the data and computation of some very basic statistical means of measure for each row or column (think of it as different cryptocurrencies) such as mean, quotient, median, standard deviation, etc. All of that is very easily achieved by that.

SciPy [16] and StatsModels [17] include some of the more advanced statistical methods, such as euclidean distance between all pairs of currencies, linear regression or testing for cointegration.

Scikit-learn [18] is suitable for more complex data transformation and algorithms usage.

Matplotlib [10] is a unique and easy to use library for plotting DataFrames into custom graphs.

Moreover, we decided to wrap all of our Python analysis into Jupyter notebook [14]. The advantages include inline plotting functionality of graphs, ability to save outputs of notebook cells (chunks of Python code), easy sharing and many more. Similarly to 4, we used git for version controlling. Together with Jupyter notebook usage, it allowed us to run and save outputs for multiple slight modifications of constants without computing over again and again and thus saving a lot of time.

5.2 Data

Importing data is simple. We define data constants (exactly the same as of the crawler - data length, quote currency and data frequency) and load the data fetched by our crawler and saved by the caching middleware. The array output of prices of the common size is perfect for importing all of our prices into one Pandas DataFrame. Let's look at the actual data and see what they look like. If not specified otherwise, we will be using 24 months of data of one hour frequency data and use Bitcoin as the quote currency in the upcoming text. That makes for 39 cryptocurrencies for our analysis

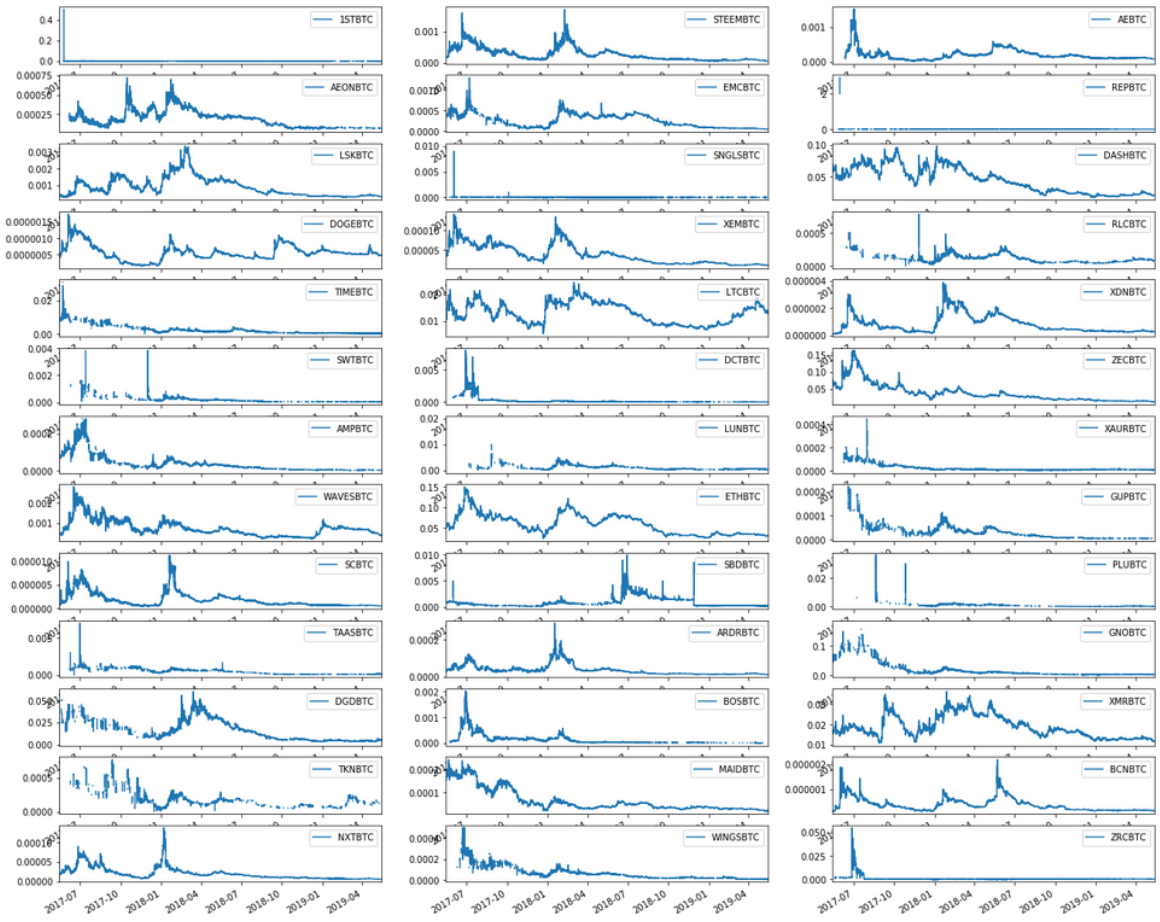

Figure 5.1: Raw fetched data. 24 months of 1 hour frequency data of all 39 cryptocurrencies we are about to analyze.

(no more currencies were available for so long on the HitBTC market, as of the date of writing this text). For easy plotting purposes we made our custom plotting function that would plot all of the cryptocurrencies' price development near each other. You can see the output in Figure 5.1.

First of all, there are graphs where we can see really little. Those are those similar to cryptocurrency *1STBTC* on the Figure 5.1. We can see that it's price was really high in the beginning when the market opened, but lowered right after. We are not really concerned about such anomalities. We want to see how the price developed and don't want to be affected by abnormally big or low values, so called outliers. To get a little better understanding, let's look into a slightly different picture, Figure 5.2. The data in the picture are the same, however, we left outliers out of the plots by limiting the view to zoom to only prices from the 5th to the 95th percentile. That way we would still see the movement of the price but neglect one or two time anomalities. Here we see that really *1STBTC* is turning down as the time progresses and not just because of the outlier in the beginning. In contrast, *ZRCBTC* seems to move quite normally, even

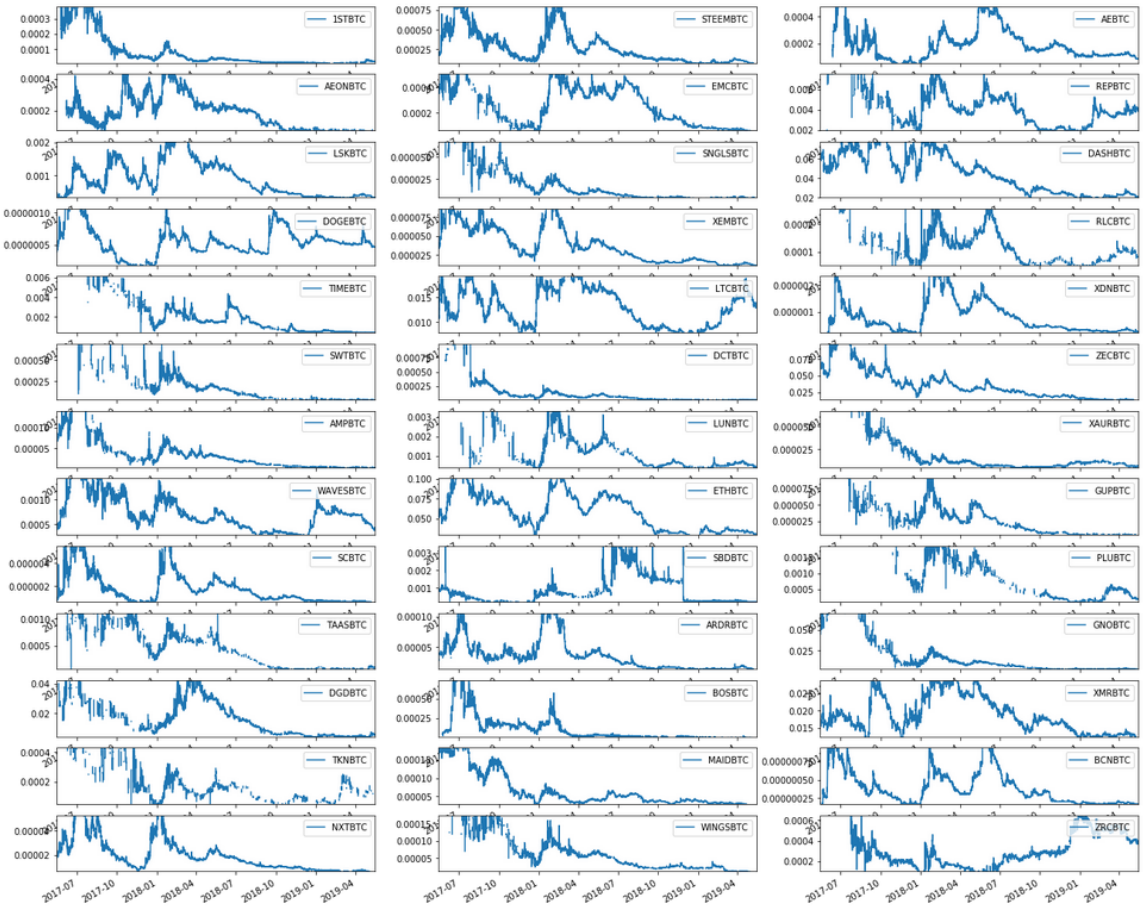

Figure 5.2: All data zoomed to the 5th till the 95th percentile of prices for each currency separately.

though we couldn't see much in the Figure 5.1. Note that we are not removing outliers from the data. We removed them just in order to better see the price development.

5.2.1 Gaps

Let's look at the Figure 5.1 for the currency pair *PLUBTC*. We can see that there is a lot of gaps indeed. As we mentioned in 4.3, gaps are left when no trade happened. We can see the statistics of how many trades happened on average in Table 5.1. For as much as half of the cryptocurrencies no trade happened for a 30% of the time. There are even currencies that recorded a trade on average for as little as 10% of the time. Now we have to choose what to do with no trade times for further analysis. Some of the standard methods include:

- **Front filling**. It stands for propagating the price from a known trade price timestamp on, until we know that another trade happened that overwrites the price. This seems natural, because the exchange actually *states* the value of the asset to be the price of the last trade, however, it may not be possible to buy or

Table 5.1: Statistics of no trade in our data. Trade ratio values represent the ratio of number of trades that **happened** relative to the length of the time frame. The 50% column represents the median value.

	min	25%	50%	75%	max
trade ratio	0.173516	0.389669	0.621575	0.982049	0.999943

sell for that price any longer. There is a chance that the trade happened by one-time selling off a big volume of asset for whatever price came to be the bid. That manipulated the price. It is normal for such events to quickly return to normal price, once the bids are restocked. However, for the purposes of this thesis we chose to ignore that and front fill the prices in order not to leave blanks that would make comovement strategies produce weird results because of the inability to correctly compute euclidean distance and other metrics.

- **Back filling**. Analogically to front filling, back filling means propagating the price backwards. This does not really make sense since the trade of that price did not happen at that time yet. Yet it may have been possible to do the trade at that price. We chose to back fill the starting prices (the only ones left blank after the front fill).

- **Ignoring**. It may be possible to ignore the timestamps when no trade happened. However if we ignore timestamps when any currency lacks price, we may get too little data. On the other hand, if we choose to ignore the timestamps lacking price pairwise, we risk exposing ourselves to many inconsistences in the comovement results. We choose rather to not ignore prices and fill them in.

In Figure 5.3 we see front filled and then back filled prices for each asset.

5.2.2 Normalizing prices

In 2.1 we described distance approach of measuring comovement and how it uses euclidean distance as its measure. However, we also mentioned that computing euclidean distance between two assets that move perfectly together, but their prices are far away from each other results into too big a distance. We suggested that our algorithm should normalize the prices first.

In short, we want to disregard the actual price the cryptocurrency has, but to keep information about the relative development of the price. In other words, we want to scale the price into a common range such that the ratio between the normalized prices of one asset to the actual prices of the asset would be constant. To do that we

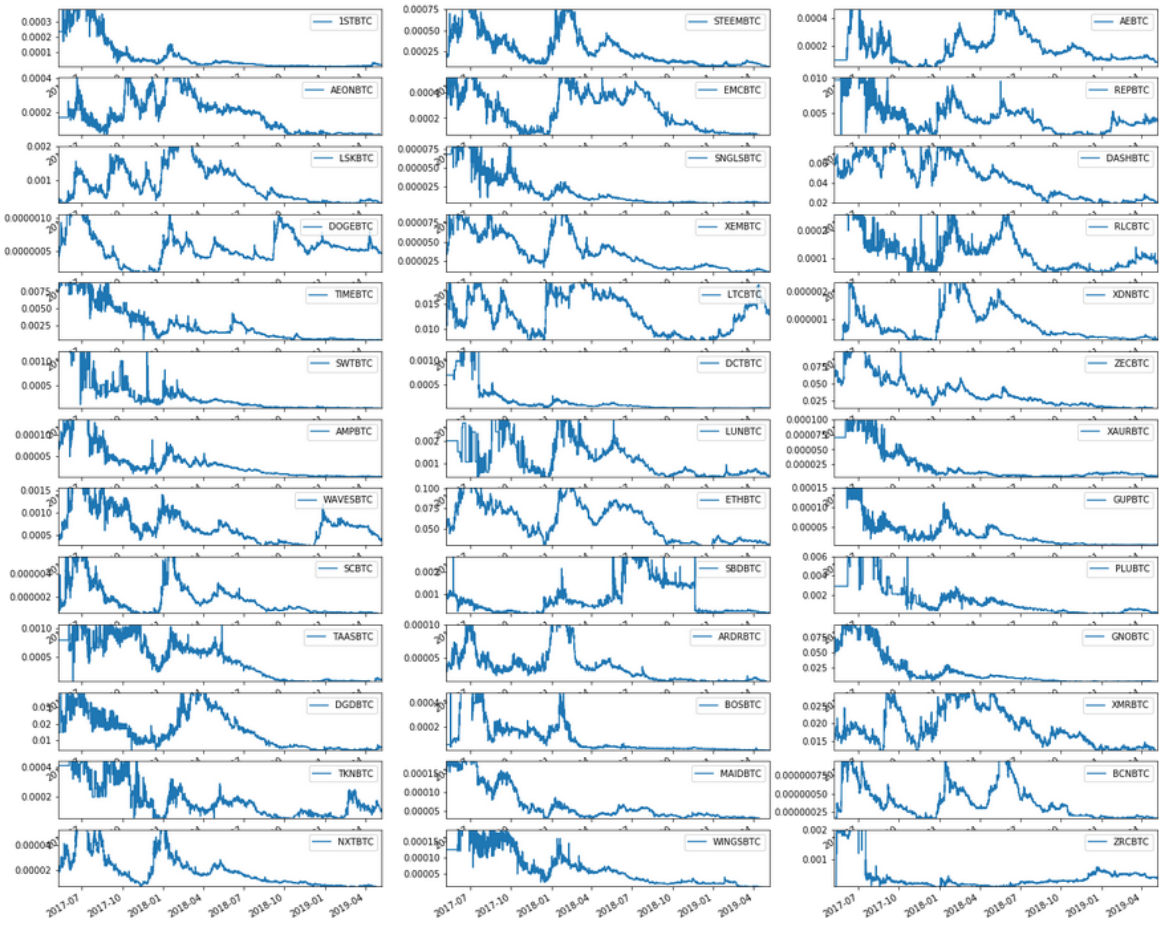

Figure 5.3: Front fill followed by back fill filled the gaps. Graphs are presented with dropped outliers.

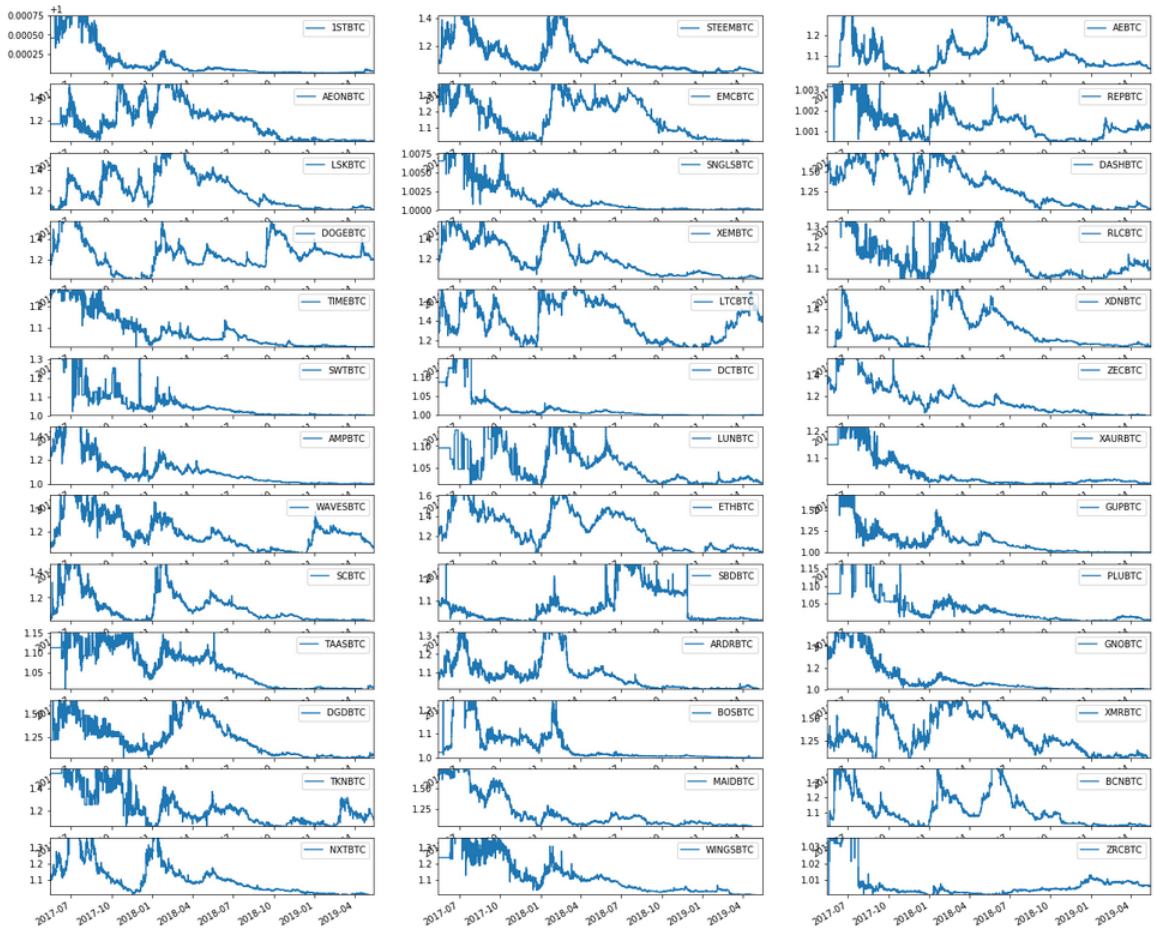

Figure 5.4: All data normalized to common range between 1 and 2, with dropped outliers.

found useful the use of Scikit-learn's MinMaxScaler [18] function that uses the following equation to normalize prices of time series (price development) of cryptocurrency X:

$$X_{std} = \frac{X - X.min}{X.max - X.min} \tag{5.1}$$

$$X_{normalized} = X_{std} \times (max - min) + min \tag{5.2}$$

Where the X_{std} stands for time series of X normalized to range $[0, 1]$. It's minimum is given value of 0, it's maximum gets value of 1 and other values are located in between by the same constant division. $X_{normalized}$ is just $[0, 1]$ normalized prices moved values to custom $[min, max]$ range. In our case, we choose $[1, 2]$ range for our normalization. We benefit from not having zero price in 6. Lastly, let's look at the normalized prices (with dropped outliers) in Figure 5.4. We see that it's the same as the graphs in Figure 5.2. The actual prices just moved to a different range.

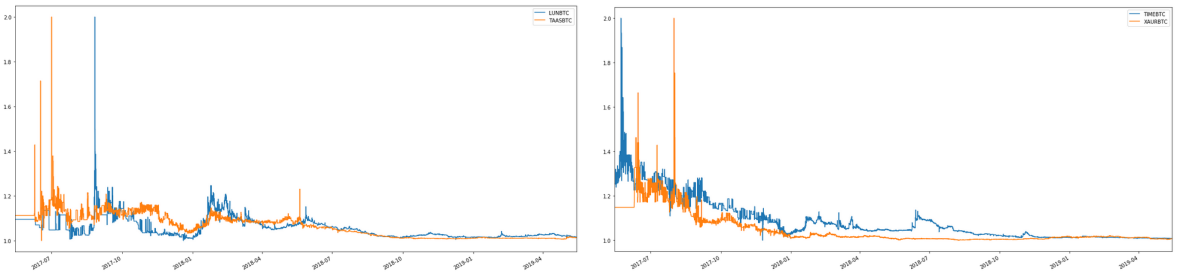

Figure 5.5: Two randomly chosen pairs from the top 20 pairs ranked by the lowest euclidean distance.

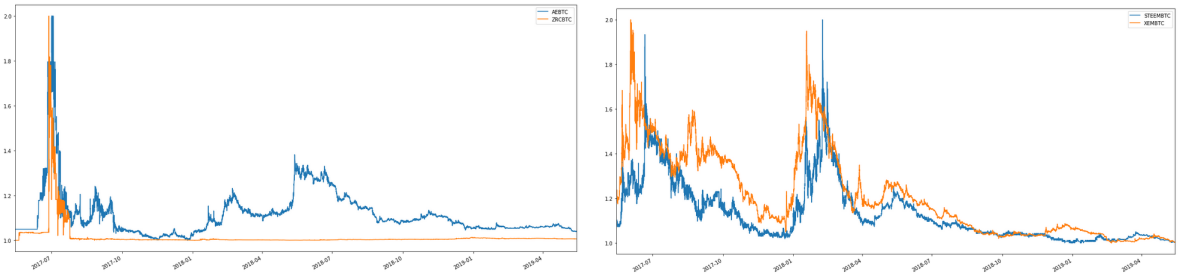

Figure 5.6: Two randomly chosen cointegrated pairs. Shown normalized, even though the cointegration test was not performed on the normalized prices.

5.3 Comovement

Now that we have data prepared, we are ready for the analysis of comovement between the currencies. For each method mentioned in 2 we explain how we computed the correlation and see some example outputs.

5.3.1 Distance approach

In 2.1 we explained how to compute the distance between two cryptocurrencies. Our implementation used existing solution by SciPy's [16] function *pdist* (pair distance) that computes pairwise euclidean distance between all pairs of currencies. We sort them by the computed distance and return. You can see two randomly chosen distance correlated pairs (in the top 20 pairs with the lowest distance) on Figure 5.5.

5.3.2 Cointegration approach

In 2.1.1 we explained what it means for a pair of cryptocurrencies to be cointegrated. For the cointegration test we used the StatsModels' [17] implementation of augmented Engle Granger statistical method included in the *coint* function call. We chose our confidence level (max pvalue) to be equal to 0.001. That means that on average 0.1% of pairs submitted for cointegration test would falsely pass. The function call returns

the pvalue. For the pair to be considered cointegrated we check whether its computed $pvalue \leq 0.001$. There is no better estimate. Pvalue is not a measure of the cointegration itself. As mentioned in [12], it may not be true that a pair with a lower pvalue is *more* cointegrated than a pair with a higher pvalue. For that reason we chose not to compare the pvalue, but rather to select a truly random selection of the cointegrated pairs to represent the approach. To compare results to the distance approach, once again you can see two randomly chosen cointegrated pairs on Figure 5.6. Note that the first cointegrated pair would have a really big euclidean distance (even normalized). Cointegration allows for a more sophisticated correlated movement.

Chapter 6

Trading

In this chapter we go into detail on how exactly we built our trading algorithm, what it's dependent on and how it responds to some market specifics.

This thesis focuses on studying comovement between cryptocurrencies. Having list of pairs of currencies whose price development is correlated, we want to use this information and suggest a trading strategy that bets on the comovement.

6.1 Methodology

We mentioned that cryptocurrency market is highly volatile in 1.3. This has both its advantages and disadvantages. The advantages include bigger profits if one knows what to do, but the disadvantages include a very high risk of losing everything. That's because the price can be manipulated by those who own a big share of the market and the whole market can crash out of a sudden in a matter of hours. However, it can spike in a matter of minutes as well. Since this movement is mostly random and hardly predictable, we aim for a strategy that is market neutral.

We already lowered our exposure to the market by choosing to trade relatively to BTC in 3.2.2. However, the above risk is still relevant. It may happen that a correlated pair is correlated because of a common vulnerable system they are built on top of. Once a vulnarability of the system is found, both currencies lose value. Bitcoin price doesn't have to decrease. It may even benefit from that. Hence we see that trading relative to BTC does not really *solve* our problem. It helps, though.

Ideally we would not want to bet on the actual currency prices. Rather we would like to bet on the relationship between the prices itself. That's possible with a trading method called **pairs trading** [12].

6.1.1 Pairs trading

Pairs trading is basically betting on a relationship between the prices of two currencies. To pairs trade, one opens two positions at the same time. One short and one long. In other words, pairs trading is about selling one currency and buying the other one. Let's illustrate that on an example:

> Let X and Y be price series of two distinct cryptocurrencies and our betting relationship equivalent to their difference $2 = Y - X$ (we aim for the price difference to be equivalent to 2). Let's create a new price series as $Y - X$. Whenever this price series spikes high enough above the threshold of 2, we expect it to revert to the mean. Hence we sell Y and buy X and wait for the relationship to revert back to normal. When it does we buy back Y and sell X back. We proceed analogically when the relationship lowers enough below the 2 threshold (buy Y and sell X).

Since we have either no or two opposite positions open at every time, we don't depend on the market. As long as the currency pairs return back to their normal relationship, we make money.

We can make money on both positions. That happens when we are lucky enough for the market to move in a direction that is good for both positions and the prices still move in order for the relationship to revert back to normal. We can lose money on X position and make money on Y or the other way around, but in that case we know that the profit would be greater. That's because the relationship returns back to normal. That means that one position had to move more than the other (they are opposite). We don't mind which one that is. We don't have to. We can even change the method used in the example to buy Y and sell X when the relationship goes up and vice versa. That's because we are not sure about which option of the mentioned happens. As long as we have zero or two opposite positions open at every moment, we are about to make money each time the relationship returns back to normal.

6.1.2 Margin trading

We mentioned that each trade consists of selling one currency and buying another one above. We are referring to the margin trading concept. That is, we can sell even without having the amount of currency. Either the exchange provides it for a fee or the exchange allows for users to offer it. It is common for the smaller cryptocurrencies to not be able to margin trade, but we are neglecting this for the purpose of this study. In margin trading, selling a currency and rebuying back for a lower price is called going short. Analogically, buying a currency and reselling for a higher price is called going long. We use the words later on.

6.1.3 When to buy

In the previous text we explained that we want to start a trade whenever a pair relationship gets far enough from the *normal* state. Next, let's look into how we can decide what exactly is far enough.

Let S be the spread of the correlated relationship. For the example used in 6.1.1, $S = Y - X$. We want to normalize that and compute what relative term *far enough* means for that corresponding spread. We followed approach that was used in [24] and [12]. They transformed the spread into standard score (also called **z-score**). We can get that by subtracting the spread's mean from the price and dividing that by the standard deviation. The spread's mean is used as the normal value around which the spread is moving up and down. The standard score then says how far the current price is from the mean, together with the direction - negative z-score means that the price is lower than the mean. On the other hand, positive z-score means higher price.

For a conservative approach, we can choose z-score thresholds ± 1.0 for initializing positions and thresholds ± 0.2 for reverting them. These values are suggested by [12].

6.2 Data split

In order for us to be able to find out how our trading algorithm actually performs with only historical data available, we decided to split the available historical data into two disjoint periods: the train and the test period. The length of the train period is twice the length of test period and it is the period which acts as an input to the methods for analyzing comovement. From that we get correlated pairs. Based on that, we try to trade during the test period. Since we do not train on the data we test on, we can get a good idea of how our algorithm performs that is useful for comparing different approaches as well as adjusting constants.

In order to not limit ourselves and to allow for a shorter term comovement trends, we added the possibility of splitting the historical dataset into multiple disjoint dataset chunks that is each split into train and test periods separately.

6.3 Algorithm

Trading itself consists of trading on several different levels:

1. **Trading the whole dataset**. To trade the whole dataset, we need to split the dataset into the desired number of chunks first. Then we perform trading on each chunk separately and sum the net profits. Since the chunks are set one after

another in time, we invest 100% of the initial capital money into each chunk trading. If we profit from one chunk, we do not reinvest the profit into the next chunk trading. We always invest the initial amount of money only. Even if we go below the capital, we invest the full capital every time. Hence the net profit from trading the whole dataset is the sum of net profits from trading the respective chunks.

2. **Trading a chunk**. To trade a chunk we need to split it into the train and test periods. Given train data only, we need to find comovement pairs based on a chosen comovement strategy from 2. We then take some comovement pairs, split the capital money in between them and simulate trading on the test dataset for each comovement pair separately. Since the capital money invested into chunk trading is divided in between the simultaneous trading of the comovement pairs, the net profit of chunk trading is determined as the average of the net profits from the corresponding comovement pairs tradings.

3. **Trading a comovement pair**. To trade a comovement pair on a given test dataset, we need to compute the pair relationship spread (based on the train dataset) and transform the spread into z-score spread (still based on train dataset only). Then we simulate trading, having at most two positions open at all times (two opposite positions, see 6.1.1). We go through the z-score spread of the test period (computed with train dataset's mean and standard deviation). If we have a position opened, we check the corresponding revert position threshold to revert the positions and take profit. If we don't, we are waiting for exceeding a placing position threshold. Based on the nature of threshold exceeded we then open or close positions. If closing a position, we collect the profit and continue trading. If we happen to be in the end of our test period and yet we have positions still open, we close them no matter the loss. Once again, we open one spread position at a time, so we compute the net profit from trading a comovement pair as the sum of net profits of the closing positions' trades. We can see sample trading of two comovement pairs on Figure 6.1. Notice the thresholds. We make money whenever it goes far away from the mean and returns back.

4. **Opening and closing spread positions**. When opening positions we just remember the actual raw prices for both currencies. When closing a position we compute the profit of the two opposite positions. We invest half of the money into one position and half into the other. Hence the net profit of the spread position is computed as the average of net profits of the respective positions. For a long position, we compute the profit as the quotient of the closing (selling) price to the opening (buying) price. For a short position, we compute the profit

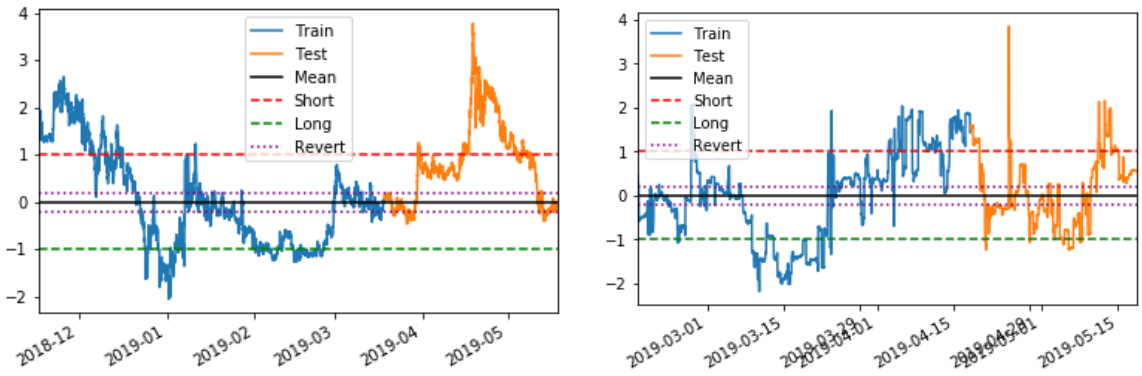

Figure 6.1: Sample trading of two comovement pairs with thresholds and periods highlighted. On the first one only one trade happened with 2.3% of net profit. On the second one five trades happened. In the end we had to perform a forced trade (the price didn't reach the revert spread position threshold) which resulted in 2.3% loss. Altogether, the pair trading was a big success, though. It made 19.6% net profit.

as the quotient of the opening (selling) price to the opening (buying) price. We compute the net profit as the profit minus one (minus the initial capital). **We suppose no trading fees**.

We implemented the trading algorithm open for modification and adjustments. The input parameters are as followed:

- **The whole dataset**. DataFrame containing all the currencies with prices.
- **Number of chunks**. The number of disjoint chunks for the whole dataset to be split into.
- **Test period size ratio**. The ratio of test period to the whole dataset size. Defaults to 1/3 (train period is twice the length).
- **Max. number of trading pairs for one chunk trading**. It may happen that so much comovement pairs are not found. This provides just an upper bound limit.
- **Function of getting the comovement pairs**. Generally one of the functions mentioned in 5.2.2. We also supply the maximum number of comovement pairs into the function, in order to select the best ones in distance approach or to halt the computation process if that much is found in cointegration approach (since we can't decide which pairs are better, we can halt as soon as we have enough pairs).
- **Function for computing the relationship spread**. In 2 we mentioned the different relationships we are betting for in different approaches. Let X and Y be

the price series. For distance approach, we just compute $Y - X$. For cointegration approach, the spread is $Y - bX$, for some $b \in \mathbb{R}$. We can compute that by running linear regression. Linear regression is conveniently included with the StatsModels [17] library.

Chapter 7

Results and discussion

In this chapter we experiment with input parameters for our trading algorithm and compare the results. We also run the algorithm on prices of different frequency. All our testing results can be found on the DVD included with this text.

Unless said otherwise, we suppose one hour data frequency of 39 currency pairs for a time frame of 24 months.

We use cumulative profit computed out-of-sample as the means of comparing results from modifications of the input parameters. We discuss input parameters for our trading algorithm in 6.3. Let's see how exactly they influence the profit. The plots come in pairs: we plot profit obtained from trading using the distance strategy with the adjusted combination of parameters on the left and the profit from trading using the cointegration strategy on the right.

7.1 Algorithm modifications

- **Risk exposure**. By risk exposure we understand the degree of trust we put into the pairs we trade. The lower the number of maximum pairs we allow to trade per chunk of dataset, the bigger our trust. If we choose to trade one comovement pair only, we put all our capital into this comovement and thus risk everything if the pair diverges. On the other hand, splitting the capital money into multiple comovement pairs eliminates the risk of a single fail. Rather it tests whether the strategy makes profit on average.

  We use statistical strategies for analyzing price comovement in the past and suppose that the same comovement holds in the future. That may and may not be true. In the cointegration comovement strategy we don't use a specific measure of how good cryptocurrencies move together. We chose a specific number of pairs that pass the comovement test. In contrast, when using the distance strategy, we

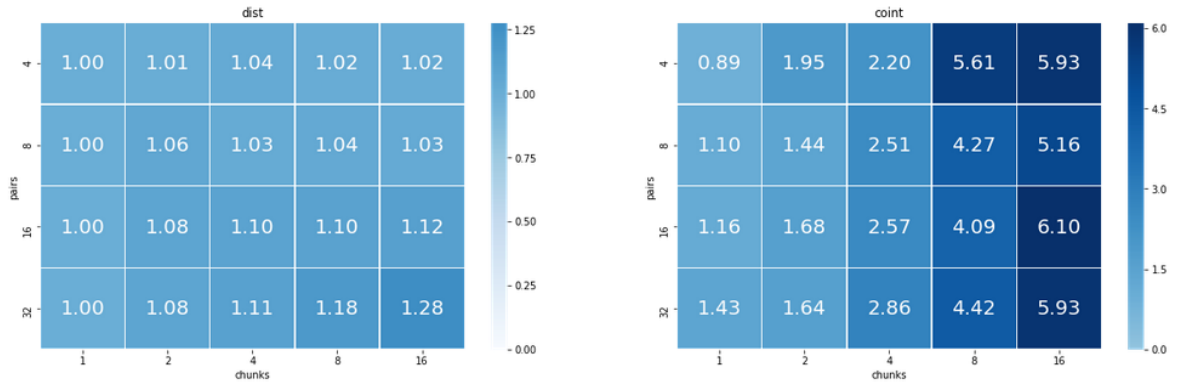

Figure 7.1: Influence of the number of chunks and max. number of pairs to trade on profit.

know exactly how far the prices are from each other and thus we have a measure of determining pairs that move *better together* than others and we select those. We can see the profit we get by limiting the maximum number of comovement pairs we trade on Figure 7.1 (the y axis). On the x axis we adjust the number of chunks, thus modifying the amount of data we train on before we trade.

If we look on the Figure column-wise, we can see that as for the cointegration approach, the results are mostly random. We suppose that's because of the random nature of selection of the comovement pairs that seem to be cointegrated. Interestingly, for the distance approach we get significantly better results when choosing to distribute the risk even though we have a way to measure the more correlated pairs. However, as [26] suggests, such results may also be the result of a too strong comovement that just seldom allows for opening positions.

- **Length of training period**. Due to the nature of crypto market, we suppose that the length of trading period proves to be a very significant parameter for the trading algorithm. There is a continuous development behind the majority of cryptocurrencies and people are increasingly familiar with the technologies behind the system. There is also a large amount of new cryptocurrencies using new technologies built on the existing ones, improving on the security, speed or any other aspect. Therefore we expect the comovement between cryptocurrencies to be rather shorter-term. We can examine that by modifying the number of chunks we split the dataset into. The profits can be seen row-wise in the same figure as before, Figure 7.1. Please note, that splitting the dataset into multiple chunks preserves the ratio of train / test timestamps, making it a valid parameter to analyze.

  We may indeed see a rather rapid increase of the earned profit. Looking at

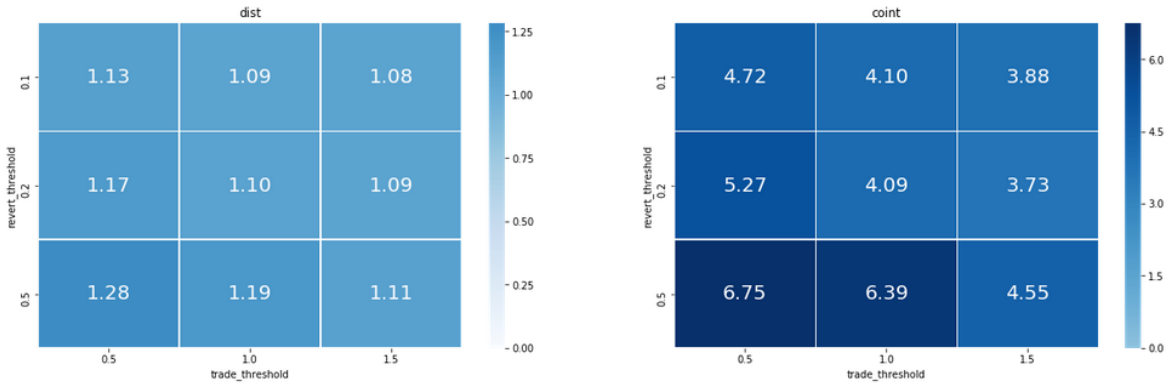

Figure 7.2: Influence of the thresholds for opening and closing positions on profit.

the distance strategy, we even do not earn anything considering the whole two years' long dataset. Splitting the dataset into chunks of a month long training period and a half a month of trading period increases the profit by 28%. For a comparison, the profit earned using the cointegration strategy goes as far as 610% of the capital investment, with the mean of 315%.

- **Trading thresholds**. In 6.1.3 we chose trading thresholds based on the suggestion by [12] that focuses on stock market trading. However, crypto market is different and as such, the used z-score thresholds of 1.0 for opening and 0.2 for closing comovement positions may prove not optimal. We considered more benevolent thresholds of 0.1, 0.2 and 0.5 for closing (y axis) and 0.5, 1.0 and 1.5 for opening positions (x axis). Results obtained from trading 16 comovement pairs on a dataset split into 8 chunks can be seen in Figure 7.2.

  We can see, that the smaller the difference between the opening and closing threshold, the bigger the profit. We suppose that we got this result mostly because of ignoring the trading fees. As such, the smaller the difference, the bigger the number of possible trades of little profit. We suggest to address and rethink this correlation in future research.

7.2 Change of frequency

In this section we are about to question the validity of presented results in terms of change of supposed frequency of data. All performed analysis so far supposed one hour price frequency. For demonstration of both lower and higher frequency, in the following text we examine fifteen minute and one day price data frequency of the same currency pairs for the same length of 24 months.

After closely looking at Figure 7.3 and Figure 7.4, we see, that our risk exposure

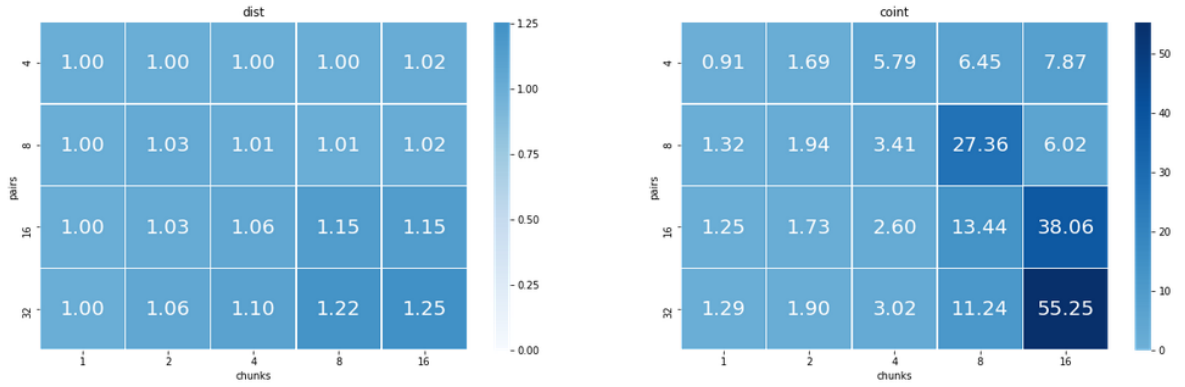

Figure 7.3: Influence of the number of chunks and max. number of pairs to trade on profit considering fifteen minute data frequency.

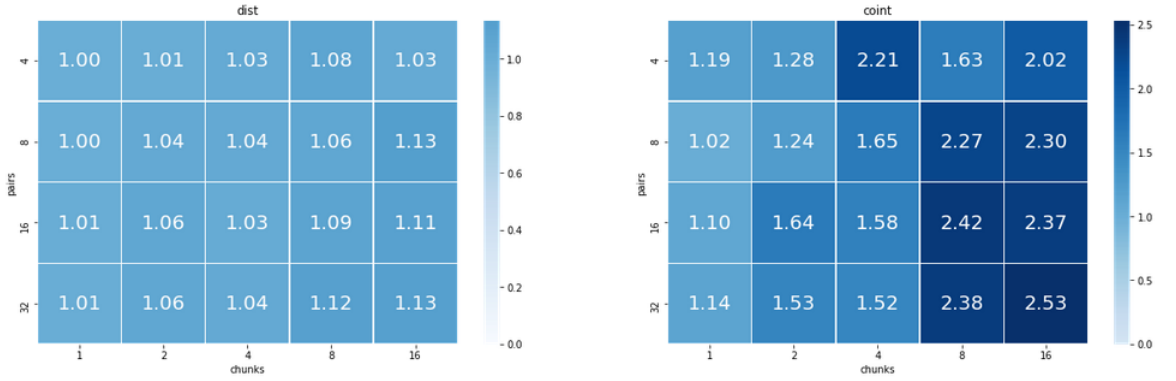

Figure 7.4: Influence of the number of chunks and max. number of pairs to trade on profit considering one day data frequency.

trend holds on average and so does the trend regarding the length of trading period. The profit tends to increase with bigger number of trading pairs both using the distance and cointegration strategy. Similarly, the profit tends to increase with bigger number of chunks. Once again, quite rapidly.

There is a bigger number of abnormalities that go against the mentioned claims in the 15 minute data compared to the hour data frequency analysis. We suppose that those abnormalities are caused too low number of trading pairs and thus higher randomness of the profit.

After looking at Figure 7.5 and Figure 7.6, we see that the overall trend is the same with fifteen minute and day frequency as it was with hour frequency. Since we do not consider trading fees, a strategy to make a higher number of smaller trades turns out to be the winner.

Once again, there are more abnormalities in the fifteen minute frequency data. The absolute winner of the cointegration strategy approach turns out to be the formerly mentioned conservative approach of 1.0 for opening and 0.2 for closing positions. We

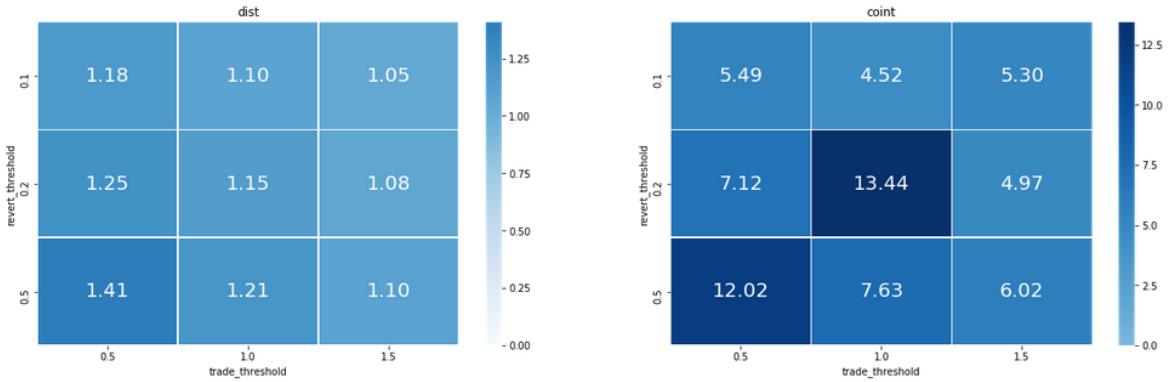

Figure 7.5: Influence of the thresholds for opening and closing positions on profit considering fifteen minute data frequency.

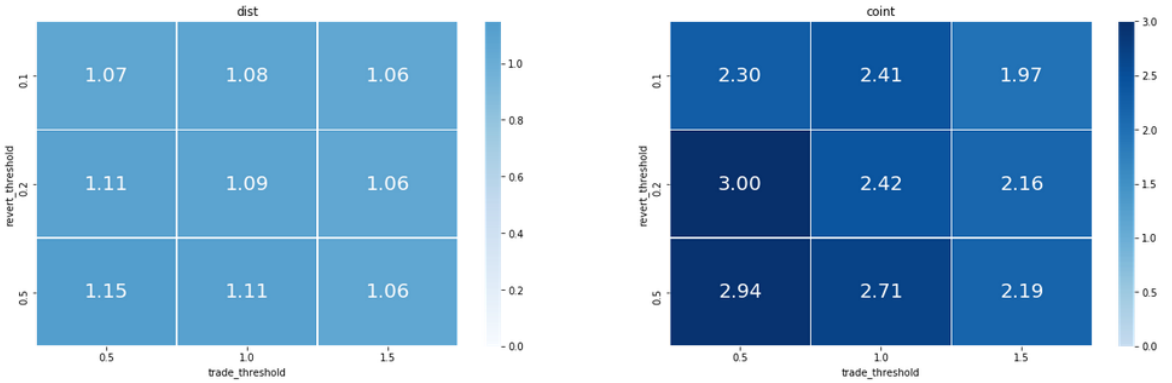

Figure 7.6: Influence of the thresholds for opening and closing positions on profit considering one day data frequency.

think that it can be the cause of more fluctuations that happen in the market with higher frequency as mentioned in 1.3.

Conclusion

We studied comovement between cryptocurrencies. Firstly, we took a look at a selection of cryptocurrencies, their purpose and the specifics of the crypto market. Then we discussed the euclidean distance and cointegration approaches as some of the well-known methods for comovement analysis commonly used in literature studying comovement on markets of different type [29], [23], [27].

In order to analyze comovement, we needed to get the actual data. We discussed various sources and removed a slight market trend by choosing to fetch prices relative to Bitcoin. We chose HitBTC [8] as the data provider and implemented a data crawler application that given the wanted data frequency, minimal data length and the quote currency gets all the available prices and saves it into a convenient format perfect for future needs. We also described the implementation details of the crawler and discussed problems we encountered.

We looked closely at two years of price data for thirty-nine cryptocurrencies of one hour frequency, preprocessed the data and provided examples of cryptocurrency pairs that move together based on the results of either euclidean distance or the cointegration strategy.

We continued to further discuss the ways of using the information in implementing a trading strategy. We ended up with a pairs trading strategy thanks to its market neutrality and found a way how to bet on the pairs comovement and nothing else. We implemented the trading strategy and described the implementation details and decision process behind.

Having left the algorithm open for modification, we compared the out-of-sample profit results of choosing the number of trading pairs, various data splits and adjusting the threshold for opening and closing positions.

We found a strong negative correlation between the length of the period for finding the comovement pairs and the consequent profit. We concluded that the comovement in the cryptocurrency market is rather short-term.

In addition to that, we found the risk distribution between multiple pairs in contrast

to betting on just a few (even better correlated) pairs to have earned bigger profit.

Across the whole study, the cointegration approach performed significantly better, earning 348% worth of net profit on average, compared to the 8% earned by the distance approach.

We supported the claims by running the trading algorithm on data of both fifteen minute and one day frequency and having found similar trends.

For the future research we suggest introducing trading fees. We expect them to influence the gains rapidly. Another interesting topic may be to convert prices to their logarithms and run distance approach on such data. We suppose that doing so would reduce rapid spikes in the prices and thus prove better for measuring comovement. For the implementation of the trading algorithm itself we also suggest to try to implement it in a rolling window fashion, re-examining the comovement-ness of the open positions and opening positions for pairs that pass the up-to-date comovement criterion.

Bibliography

[1] About node.js, May 2019. https://nodejs.org/en/about/.

[2] Babel is a javascript compiler, May 2019. https://babeljs.io/.

[3] The best way to learn modern javascript, May 2019. https://es6.io/.

[4] Coin market cap, Feb 2019. https://coinmarketcap.com.

[5] Cryptomiso, May 2019. https://www.cryptomiso.com/.

[6] Eslint: The pluggable linting utility for javascript and jsx, May 2019. https://eslint.org/.

[7] git, May 2019. https://git-scm.com/.

[8] Hitbtc api, May 2019. https://api.hitbtc.com/.

[9] How to make money on arbitrage with cryptocurrencies, Feb 2019. https://hackernoon.com/how-to-make-money-on-arbitrage-with-cryptocurrencies-6618bdad3ce1.

[10] Matplotlib: Python plotting, May 2019. https://matplotlib.org/.

[11] npm.js, May 2019. https://www.npmjs.com/.

[12] Online lectures in statistical and financial topics, May 2019. https://www.quantopian.com/lectures.

[13] Prettier: Opinionated code formatter, May 2019. https://prettier.io/.

[14] Project jupyter, May 2019. https://jupyter.org/.

[15] Python data analysis library, May 2019. https://pandas.pydata.org/.

[16] Scipy library, May 2019. https://www.scipy.org/scipylib/index.html.

[17] Statsmodels: Statistics in python, May 2019. https://www.statsmodels.org/stable/index.html.

[18] Statsmodels: Statistics in python, May 2019. https://scikit-learn.org/stable/.

[19] Top 100 cryptocurrency exchanges by trade volume, Feb 2019. https://coinmarketcap.com/rankings/exchanges/.

[20] Welcome to python.org, May 2019. https://www.python.org/.

[21] What is a cryptocurrency, Feb 2019. https://www.investopedia.com/terms/c/cryptocurrency.asp.

[22] What is monero, Feb 2019. https://ww.getmonero.org/get-started/what-is-monero/.

[23] August Abelvik-Engmark and Daniel Vårdal Haugland. Spread trading in brent crude futures. *Norwegian University of Science and Technology*, 120:1–120, 2018.

[24] João Caldeira and Guilherme V. Moura. Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy. *SSRN*, 28:1–28, 2013.

[25] Paraskevi Katsiampa. Volatility co-movement between bitcoin and ether. *Finance Research Letters*, 2018.

[26] Fredrik Kirkestuen and Christian Thomassen. Statistical arbitrage using high frequency pairs trading. *Oslo Business School at Oslo Metropolitan University*, 37:1–37, 2018.

[27] Claudio Morana and Andrea Beltratti. Comovements in international stock markets. *Journal of International Financial Markets, Institutions and Money*, 18(1):31–45, 2008.

[28] David Z. Morris. Tether now admits it's not fully backed by dollars. *BREAKERMAG*, 2019. https://breakermag.com/tether-now-admits-its-not-fully-backed-by-dollars/.

[29] Robert S Pindyck and Julio J Rotemberg. The excess co-movement of commodity prices, 1988.

[30] John W. Tukey. *Exploratory Data Analysis*. 1977. First edition.