FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Bachelor's Thesis

High Frequency Currency and Cryptocurrency Trading

Max Nonfried

PILSEN, CZECH REPUBLIC

2023

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Bachelor's Thesis

High Frequency Currency and Cryptocurrency Trading

Max Nonfried

Thesis advisor
Ing. Jan Pospíšil, Ph.D.

PILSEN, CZECH REPUBLIC 2023

Citation in the bibliography/reference list:
NONFRIED, Max. *High Frequency Currency and Cryptocurrency Trading*. Pilsen, Czech Republic, 2023. Bachelor's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Ing. Jan Pospíšil, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Max NONFRIED**
Osobní číslo:	**A19B0601P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Vysokofrekvenční obchodování s měnami a kryptoměnami**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznámit se se základními principy algoritmického vysokofrekvenčního obchodování s měnami a kryptoměnami.
2. Zpracovat podrobnou rešerši metod detekce a predikce různých režimů v časových řadách vysokofrekvenčních finančních dat.
3. Popsat základní vlastnosti zvolených metod a způsobů jejich aplikace na zkoumané finanční časové řady včetně návrhu obchodní strategie.
4. Ve vhodném programovacím jazyce implementovat tyto metody tak, aby je bylo možné použít pro algoritmické obchodování.
5. Provést srovnání jednotlivých metod a úspěšnosti navržených obchodních strategií za použití reálných dat.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Jan Pospíšil, Ph.D.**
Katedra matematiky

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Declaration

I hereby declare that this Bachelor's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

Pilsen, on 25 April 2023

. . . . . .

Max Nonfried

Abstract

The main focus of this thesis is on the utilization of classical statistical methods, specifically AR, MA and ARIMA models, with high frequency trading data. We consider two datasets: the first with quotes of the currency pair EUR/USD and the second with trades of the cryptocurrency pair BTC/USDT. The frequency of the data is several values per second in both cases. In the theoretical part of the thesis, there is a detailed description of the mathematical and statistical foundations for the proper use of the ARIMA model and practical information related to trading principles. The practical part introduces the description of data processing, including analysis of the statistical distribution of data, and furthermore, the comparison of a trading strategy based on the forecast from the ARIMA model and two basic trading strategies. The practical part also contains an overview of the so called backtesting of the trading strategies.

Abstrakt

Tato práce se zaměřuje na využití klasických statistických metod, konkrétně AR, MA a ARIMA modelů, aplikovaných na vysokofrekvenční obchodní data. Uvažujeme dvě datové sady: první sada obsahuje cenové nabídky na měnovém páru EUR/USD, druhá sada obsahuje realizované obchody na kryptoměnovém páru BTC/USDT. Frekvence těchto dat je několik hodnot za sekundu. V teoretické části je podrobný popis matematických a statistických podkladů pro správné použití ARIMA modelu a praktických informací ohledně obchodních principů. Praktická část popisuje zpracování dat, zahrnuje analýzu týkající se statistického rozdělení dat a dále uvádí porovnání strategie založené na předpovědi z ARIMA modelu se dvěma základními obchodními strategiemi. Praktická část také obsahuje stručnou rešerši tzv. zpětného testování obchodních strategií.

Keywords

High frequency trading • ARIMA model • Currency • Cryptocurrency • Backtesting

Acknowledgement

I would like to express my sincere gratitude to Ing. Jan Pospíšil, Ph.D. for his unwavering guidance and support. His expertise and patience have been invaluable to me and helped me to write this thesis.

Contents

Glossary and notation

$\mathbb{N}$ Natural numbers.

$\mathbb{R}$ Real numbers.

$\mathbb{Z}$ Integers.

AR Autoregressive model.

ARIMA Autoregressive Integrated Moving Average model.

ARMA Autoregressive Moving Average model.

IaT Interarrival Times.

MA Moving Average model.

Introduction 1

Since the establishment of the first stock exchanges, people have been speculating on the price of assets in order to make money. The intent of speculation is still the same; however, the form of trading changes over time. The last substantial change came with the evolution of computer technology. Besides traditional traders, automatic trading algorithms have become part of the market. The forecasting of price movement with the usage of computer algorithms and statistical methods has been the object of much research over the last decades. The matter, related to the thesis, that changed a lot is the frequency of executing trades. With the use of automatic algorithms, the frequency can reach several trades per second. In this thesis, we work specifically with two datasets of such frequency; the first contains quotes on the currency pair EUR/USD, and the second contains executed trades on the cryptocurrency pair BTC/USDT.

There are many methods of forecasting the behavior of an asset. Techniques of the time series analysis containing statistical models such as ARIMA or GARCH models, usage of Hidden Markov Models Lunga, 2020, or the modern applied data science methods based on machine learning. The main focus of the thesis is on the classical statistical methods, mainly exploiting the AR, MA, ARIMA models and moving averages. We compare two basic trading strategies, founded on moving averages, with the strategy founded on the forecast from the ARIMA model. In order to properly choose the tool for testing the strategies against each other, we provide a brief overview on backtesting frameworks in the programming language Python.

There exist several books considering trading with data with the high frequency (High frequency trading) as Cartea; Jaimungal; Penalva, 2015, Degiannakis; Floros, 2015. The mathematical foundation of time series analysis is explained in the book Cipra, 2020. The book Halls-Moore, 2016 provides a description of trading strategies, mathematical foundations of statistical methods, and also a general context. The ARIMA models are explained in Nau, 2020, Spyros G. Makridakis, 1998 in a more popular way. Though trading deals heavily with statistical methods and strategies, there are many important practical details involved. Garner, 2012 provides a

complex view of the structure of the trading system that encompasses all tenets of the process.

The structure of the thesis is as follows. In Chapter 2, we introduce the basic mathematical, statistical, and practical foundations for this thesis. In Chapter 3, we describe the process of backtesting the trading strategies, the research on the statistical distribution of data, and the structure and principles of trading strategies. In Chapter 4, we describe the processing of datasets in detail. Furthermore, we introduce the results of fitting the statistical distribution and ARIMA model to data. Lastly, we write about the results of testing the trading strategies. We conclude the thesis in Chapter 5.

Preliminaries 2

2.1 High frequency trading

Defining *high frequency trading* (HFT) is a difficult and challenging task. Regarding research Degiannakis; Floros, 2015, p. 4-7, there is not one generally accepted uniform definition. It depends on the author and the year of the research (older versus newer), from which perspective one looks, etc.

The main characteristics which are held in common across definitions and which have a rather general character include among others: HFT uses sophisticated algorithms, the term HFT is a subsection of algorithm trading, HFT uses computer technology, HFT has a very high number of orders, HFT has a short time of holding position. Degiannakis; Floros, 2015, p. 4-7

Most of the differences in definitions lie in diverse specifications of some characteristics mentioned above; for example, how long a person holds trading position. This varies from milliseconds to hours, in some definitions to days. Another difference is which strategies/algorithms are used for HFT. Algorithms can be based on classical trading principles just customized for HFT, or newly designed only for HFT. Degiannakis; Floros, 2015, p. 4-7

For purposes of this thesis, taking into account the nature of the thesis, we chose the statement of Office of the Federal Register; Records, 2010, p. 281 on characteristics of HFT as the best-matched definition:

1. The use of extraordinarily high-speed and sophisticated computer programs for generating, routing, and executing orders;

2. use of co-location services and individual data feeds offered by exchanges and others to minimize network and other types of latencies;

3. very short time-frames for establishing and liquidating positions;

4. the submission of numerous orders that are canceled shortly after submission; and

5. ending the trading day in as close to a flat position as possible (that is, not carrying significant, unhedged positions overnight).

Considered frequencies of our data will be specified below in section 2.3.

2.2 Currency and cryptocurrency

A *currency* denotes a national form of money. The theoretical definition of money considers money as any asset which is generally accepted for payments for goods, services, or debts. An important note is that each currency has characteristics enshrined in the law. Revenda, 2014, p. 14, 19, 20

Cryptocurrencies are a matter of recent years. Therefore, the formal definitions and laws related to cryptocurrencies are in the process of formation. The first of the two following paragraphs contains rather a crypto-community point of view, the second rather official.

Wikipedia defines cryptocurrency as "a digital currency designed to work as a medium of exchange through a computer network that is not reliant on any central authority, such as a government or bank, to uphold or maintain it,"[1] it is based on asymmetric cryptography and decentralization. Taking Bitcoin as an example of cryptocurrency, the portal *bitcoin.org* states that Bitcoin is "a consensus network that enables a new payment system and a completely digital money"[2] and *Bitcoin Wiki* states that "bitcoin is a decentralized digital currency that enables instant payments to anyone, anywhere in the world."[3]

Despite these assertions, the cryptocurrencies are not money in the economic or law meaning according to statement of the Czech National Bank [4]. The similar stance is taken also by other national banks mentioned in the declaration.

2.3 Financial time series

Cipra, 2020, p. 5 defines time series as "any sequence $y_1, \ldots, y_n$ ordered chronogically in time". In the thesis, we will usually denote time series as $X = (x_t, t \in \mathbb{N})$.

[1] Wikipedia contributors. Cryptocurrency. *Wikipedia, The Free Encyclopedia* [online]. Last updated 29.3.2023. [Cit. 30.3.2023]. Available at: `https://en.wikipedia.org/wiki/Cryptocurrency`

[2] FAQ. *bitcoin* [online]. [Cit. 30.3.2023]. Available at: `https://bitcoin.org/en/faq#general`

[3] *Bitcoin Wiki* [online]. Date of last revision 9.2.2021. [Cit. 30.3.2023]. Available at: `https://en.bitcoin.it`

[4] Je k obchodování s tzv. převodními tokeny nebo k jejich směně vyžadováno oprávnění ČNB?. *Czech National Bank* [online]. Prague: Czech Natioal Bank, 19. 11. 2018. [Cit. 30.3.2023]. Available at: `https://www.cnb.cz/cs/dohled-financni-trh/legislativni-zakladna/stanoviska-k-regulaci-financniho-trhu/RS2018-13/`

Further, Cipra, 2020, p. 199 writes that financial time series "are usually derived from prices P_t of financial assets at time (e.g., stocks or commodities, but also indices) and price variations $\Delta P_t = P_t - P_{t-1}$". We work with price variation of logarithmic returns (see subsection 2.6.7). Our financial time series is currency tick data. The more detailed general description of tick data is in subsection 2.8. In section 4.1, there is a description of our data. Our data frequency is several values per second.

2.4 Statistical models

For better orientation in the placement of methods and models used in this thesis, we briefly introduce considered general classification distinguishes:

- parametric and non-parametric models,
- conditional and unconditional models.

It is worth mentioning that besides statistical models, there are other models with the power to make predictions as well. Broadly we can call them predictive models. They include for example some models from machine learning such as decision trees, boosted trees, neural networks, support vector machines, etc. Taboga, 2021

1. Parametric models. Their behavior is defined by the set of parameters, which are the only information needed to make predictions. An example of a parametric model is linear regression. Halls-Moore, 2016, p. 223

2. Non-parametric models. Non-parametric models do not contain any parameters (in the meaning of parametric models parameters) Halls-Moore, 2016, p. 223. They can be directly analyzed with techniques for example multivariate kernel density estimation or kernel regression Taboga, 2021. In comparison with parametric models, the non-parametric models have benefits in the form of their greater flexibility. On the other hand, non-parametric models need to have a big amount of training data. In the financial field, there is also a higher risk of overfitting the model Halls-Moore, 2016, p. 223.

3. Conditional models. Taboga, 2021 states that "in conditional models (also called discriminative models), the sample is partitioned into input and output data, …. The statistical model is obtained by placing some restrictions on the conditional probability distribution of the outputs given the inputs". The conditional models have two classes:

    - In *regression models* is the output variable continuous. An example of a regression model is linear regression (an assumption of a linear relation

between input and output) or non-linear regression (non-linear relation between input and output).

- On the contrary, *classification models* have the output variable in a discrete form. The examples are the logistic classification model (also called logit model) and the multinomial logit model.

4. Unconditional models. Taboga, 2021 writes that the unconditional models (also called generative models) are "used to analyze the joint distribution of inputs and outputs".

The main focus of the thesis will be on the *Autoregressive Integrated Moving Average* model (ARIMA), *Autoregressive* model (AR), and *Moving Average* model (MA). They are commonly used models in time series analysis. According to the classification, they are classified as parametric, conditional models.

2.5 ARIMA model

ARIMA is an abbreviation for Autoregressive Integrated Moving Average model. It is a statistical predictive model, partially linear regression, and it is classified as a parametric, conditional model (see section 2.4). It was popularized by statisticians George Box and Gwilym Jenkins in the 1970s. Thus the term *Box-Jenkins model* or *Box-Jenkins methodology* refers to the ARIMA model. The ARIMA model can be seasonal or nonseasonal; in this thesis, we work with nonseasonal. Information about the ARIMA model in this section comes mostly from the source Nau, 2020 and Spyros G. Makridakis, 1998.

As the name ARIMA suggests, the model joins three separate processes: Autoregressive (AR), Integrated (I), and Moving Average (MA). Each of these processes can stand separately. The autoregressive process and moving average process are AR and MA models. The integrated process is a differencing of time series with the aim to achieve stationarity of the series. The case of ARIMA$(p, 0, 0)$ is equal to model AR(p), case of ARIMA$(0, 0, q)$ is equal to model MA(q). The connection of AR and MA model is called *Autoregressive Moving Average* model (ARMA). The definition of AR and MA processes follows.

2.5.1 AR process

AR(p) process of order p is a linear combination of past observations (values) with relation:

$$X_t = \alpha_1 X_{t-1} + \ldots + \alpha_p X_{t-p} + \epsilon_t. \tag{2.1}$$

AR terms capture a mean reversion and momentum effect of a series. When AR has $p > 1$, the sum of its coefficients determines the speed of mean reversion. When AR

has $p = 1$, it holds that the closer to zero coefficient is, the faster the series returns to its mean; the closer to 1, the slower it returns to its mean.

2.5.2 MA process

MA(q) process of order is a linear combination of past error terms with the formula:

$$X_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \ldots + \beta_q \epsilon_{t-q}. \tag{2.2}$$

MA terms capture random shocks in a series. It expresses the error between the fitted model and the real values. The MA(q) coefficient is a part of the shock in the q period, which can be still noted in the current period.

2.5.3 Integration process

The integration process stationarizes the series by differencing (math definition is written in the subsection below 2.5.4). It removes trends, time-various trends, etc. It is possible for data to exhibit a strong trend that differencing can not capture, in which case it is necessary to consider another method to make the series stationary (e.g. a logarithmic transformation). The stationary series is what we need for an ARIMA model and it is the first step when fitting an ARIMA model.

2.5.4 Mathematical definition of ARIMA

The ARIMA model has three parameters p, d, q, that define the model ARIMA(p, d, q). The number of AR terms is denoted by p, the number of differences by d, and the number of MA terms by q. An ARIMA($p, 0, d$) model is the same as an ARMA(p, d), it is a stationary process X_t with relation as follows:

$$X_t = \mu + \alpha_1 X_{t-1} + \ldots + \alpha_p X_{t-p} + \epsilon_t + \beta_1 \epsilon_{t-1} + \ldots + \beta_q \epsilon_{t-q}, \tag{2.3}$$

where μ is a constant, α are the parameters for the AR process, β are the parameters for the MA process, and ϵ is a white noise process (errors).

The relation above can be used for $d = 0$ or on series after differencing. A formula for differencing process, where X denotes the original series and x denotes the differenced series, is:

$$\begin{aligned} &d = 0 : x_t = X_t, \\ &d = 1 : x_t = X_t - X_{t-1}, \\ &d = 2 : x_t = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}. \end{aligned} \tag{2.4}$$

We can write higher differences more conveniently with a backward shift (backshift) operator B, which shifts by multiplying the observation back in time by 1 period:

$$\begin{aligned} BX_t &= X_{t-1}, \\ B^2X_t &= B(BX_t) = B(X_{t-1}) = X_{t-2}, \\ B^nX_t &= X_{t-n}. \end{aligned} \tag{2.5}$$

Now with the knowledge of the backshift operator, we can define the complex relation for the ARIMA process including differencing:

$$\alpha_p(B)(1-B)^dX_t = \beta_q(B)\epsilon_t. \tag{2.6}$$

2.6 Time series properties

2.6.1 Stationarity

We commonly distinguish between two types of *stationarity*: strict and weak. In the thesis, the term stationarity refers to weak stationarity. Stationarity of time series means that statistical properties (mean, variance, covariance, etc.) do not change over time. So there is incompatibility with situations when data has a trend or seasonality character. A lot of forecasting methods work only with stationary data. Therefore the data are often transformed from non-stationary to stationary. A series X_t is stationary if for every s and t follows the relations:

$$E(X_t) = \mu, \tag{2.7}$$

$$\operatorname{cov}(X_s, X_t) = E(X_s - \mu)(X_t - \mu) = \operatorname{cov}(X_{s+h}, X_{t+h}), \tag{2.8}$$

where $\mu \in \mathbb{R}$ is constant, $h \in \mathbb{R}$ is arbitrary, and

$$\operatorname{var}(X_t) = \operatorname{cov}(X_t, X_t) = \sigma_X^2, \tag{2.9}$$

where $\sigma_X^2 \geq 0$ is constant. Cipra, 2020, p. 124

2.6.2 Autocorrelation

Autocorrelation Function (ACF), also called serial correlation, gives us information about the correlation between two observations at different times Cipra, 2020, p. 125. In time series analysis, we use the autocorrelation function to identify the lags with significant correlation. This information is important to properly model the time series data. Definition of autocorrelation function ρ_k of lag k:

$$\rho_k = \frac{\operatorname{cov}(x_t, x_{t-k})}{\sigma_x^2}, \qquad k \in \mathbb{Z}. \tag{2.10}$$

The name of the autocorrelation function graph is *correlogram*.

2.6.3 Partial autocorrelation

Besides the autocorrelation function, *Partial Autocorrelation Function* (PACF) is also used in time series analysis. PACF is used especially for identifying the order of AR, ARMA, and ARIMA models. Nau, 2020 writes that the PACF "shows the amount of autocorrelation at lag k that is not explained by lower-order autocorrelations". The mathematical definition is not trivial but rather complex, thus we introduce only recursive *Durbin–Levinson algorithm* that is used in practise to calculate PACF, denoted as $\phi_{k,k}$, for $n \geq 1$:

$$\phi_{n,n} = \frac{\rho(n) - \sum_{n-1}^{k=1} \phi_{n-1,k}\rho(n-k)}{1 - \sum_{n-1}^{k=1} \phi_{n-1,k}\rho(k)}, \tag{2.11}$$

where, for $n \geq 2$

$$\phi_{n,k} = \phi_{n-1,k} - \phi_{n,n}\phi_{n-1,n-k}, \qquad k = 1, 2, \ldots, n-1, \tag{2.12}$$

and $\rho(n)$ denotes the autocorellation function Robert H. Shumway, 2017, p. 104. The whole mathematical definition of a partial autocorrelation function is in the book Cipra, 2020, p. 127.

2.6.4 White noise

In time series analysis, the series which has serially uncorrelated variables, zero mean, and constant variance $\sigma^2 > 0$ is called white noise Cipra, 2020, p. 11. In some model selection processes, we want the residual component of the time series to be the white noise Halls-Moore, 2016, p. 89.

2.6.5 AIC, BIC

Akaike Information Criterion (AIC) and *Bayesian Information Criterion* (BIC) or also *Schwarz information criterion* are criteria for model selection. We can use them for example when we are searching for the right order of the ARIMA model. They are based on the idea: "…penalize unnecessarily high orders k and l (simultaneously with the minimization of residual standard deviation) achieving the consistency of the estimates in this way" Cipra, 2020, p. 140. The lower AIC or BIC the model has, the better model it should be. The formulas of AIC and BIC follow:

$$AIC(k,l) = \ln \widehat{\sigma}^2_{k,l} + \frac{2(k+l+1)}{n}, \tag{2.13}$$

$$BIC(k,l) = \ln \widehat{\sigma}^2_{k,l} + \frac{(k+l+1)\ln n}{n}, \tag{2.14}$$

where k and l are the unnecessarily high orders of ARMA(k,l), $\widehat{\sigma}^2_{k,l}$ is the estimated variance of white noise in the process ARMA(k,l), and n is the length of the series.

2.6.6 Ljung-Box test

Ljung-Box test is a statistical test that examines m autocorrelations of time series. We use the test on residuals of time series to determine if a good fit of the model was achieved. If the autocorrelations are small or zero we can conclude that a good fit was achieved.

The hypotheses are:

- H_0: the data are independently distributed,
- H_a: the data are not independently distributed.

The test statistic is:

$$Q = n(n+2)\sum_{k=1}^{m}\frac{\hat{\sigma}_k^2}{n-k}, \tag{2.15}$$

where n is the length of the time series, m is the number of tested lags and $\hat{\sigma}_k$ is the estimated autocorrelation at lag k. The critical region of significance level α for rejecting the H_0 hypothesis is:

$$Q > \chi^2_{1-\alpha,h}, \tag{2.16}$$

where $\chi^2_{1-\alpha,h}$ is the chi-square distribution table value with h degrees of freedom; $h = m - p - q$, where p, q is the order of for example ARMA(p, q) model.[5]

2.6.7 Logarithmic returns

The usage of relative prices for analyzing lightens the comparison between different assets or monetary units. Therefore in financial practice, it is usual to work with prices P_t in form of logarithmic returns (log returns) r_t. Cipra, 2020, p. 161, 199 Log returns follow the relation:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln\left(P_t\right) - \ln\left(P_{t-1}\right). \tag{2.17}$$

Another benefit of log returns is that change in natural logarithm is approximately equal to percentage change. This approximation is valid only for small changes. Nau, 2020 The mathematical definition follow:

$$r_t \approx \frac{P_t - P_{t-1}}{P_{t-1}}. \tag{2.18}$$

The differences between equality and approximation for small and big changes are in Table 2.1.

[5] *NIST/SEMATECH e-Handbook of Statistical Methods* [online]. [Cit. 10.1.2023]. Available at: `https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4481.htm`

Table 2.1: Comparison between price changes in % and in log difference
Nau, 2020

Change in P in %	Log difference of P
-50%	-0.693
-40%	-0.511
-30%	-0.357
-20%	-0.223
-10%	-0.105
-5%	-0.051
-2%	-0.020
0%	0.000
2%	0.020
5%	0.049
10%	0.095
20%	0.182
30%	0.262
40%	0.336
50%	0.405
100%	0.693

2.7 Other statistical methods

There are other statistical methods or tools mentioned in the thesis. Because they are not the major foundations for the thesis, we will not introduce them here. Instead, we give the list of references to statistical books in which there are exact and thorough definitions.

- Kolmogorov-Smirnov test (KS), Dodge, 2008, p. 284, is a non-parametric goodness-of-fit test. One-sample KS test is used to decide if a sample follows the hypothesized probability distribution. With a two-sample KS test, we can determine whether the distributions of the two samples differ.

- Least-squares method, Dodge, 2008, p. 305, is a method for minimizing the sum of squared residuals. It is an important approach in data fitting.

- Augmented Dickey-Fuller test, Cipra, 2020, p. 154, is a statistical test if the series has a unit root. In time series analysis is used to determine whether the series is stationary.Tsay, 2005, p. 68, 69.

2.8 Description of tick data

The tick data (tick-by-tick data) is often defined as a data which contains information (timestamp, price, volume) about each executed trade or bid/ask price quote. This is true on a general scale. In most cases, the data providers specify what their data relates to and which information the data contains. It could be for example the following possibilities:

- trade tick data – data containing the information about each executed trade;
- bid/ask data (1) – data containing the information about best bid/ask quotes;
- bid/ask data (2) – data containing the information about best bid/ask quotes and about best bid/ask quotes of each registered dealer;
- bid/ask data (3) – data available only to member companies of NASD (National Association of Securities Dealers) and are connected with execution and amendment privileges.[6]

Note: in trading, the term *tick* also denotes minimal possible price change. The size of the tick depends on regulations and varies according to the type of trading instrument.

2.9 Prediction

In the thesis, a prediction of time series is generated by statistical models. The prediction can be in-sample or out-of-sample. The in-sample prediction predicts the same data which was used to fit the model. Out-of-sample prediction predicts values that were not used to fit the model. For example, we have data for the years 2000-2023. We fit the model to data 2000-2020. The out-of-sample prediction would be for the years 2021-2023, and the in-sample for the years 2000-2020. Of course, with the out-of-sample prediction we can also predict values that we do not know yet, e.g. the year 2024. Cipra, 2020, p. 19

Besides the term *prediction*, there is also a term *forecast*. Although some authors use the terms differently, in this thesis, they are used interchangeably.

[6] Historical Tick Data. *FirstRate Data* [online]. [Cit. 15.3.2023]. Available at: `https://firstratedata.com/tick-data`

Methodology 3

3.1 Implementation

The thesis's practical part includes source codes written in programming language *Python*. The codes are written in the form of short scripts with lengths mostly from 10 to 100 rows. There is a list of the scripts in the appendix 5 containing a description of every single script. The scripts are written in Python of version 3.9.13 and can be executed with the command

```
python file_name.py
```

in *PowerShell* and *Command Prompt* using operating system *Windows*. The Jupyter notebooks can be opened as follows:

1. open PowerShell in the directory where there are notebooks,
2. execute the command

   ```
   python -m notebook
   ```

   in PowerShell,
3. copy the generated path to any internet browser.

3.2 Python

Python is a high-level, multi-paradigm programming language. In recent years it has become very popular. In the data science field, it is the main competitor of the language *R*. It is highly extensible via modules and its philosophy is based on "friendliness" towards programmers.

For the purposes of the thesis, we use mostly four packages.

- For data manipulation and analysis we use package `Pandas`[1].
- Package `SciPy`[2] provides algorithms for scientific computing. Among others, it has the module `stats`, which contains some parts of statistics, for example, probability distributions and statistical tests.
- Statistical models (AR, ARIMA, ...) are implemented in the package `statsmodels` [3].
- `Matplotlib`[4] is a package for creating graphical plots. Another good package for plots is `Seaborn`[5], the package built on `Matplotlib`, but the outputs have a more professional look.

Some codes are written in *Jupyter Notebook*[6]. Jupyter Notebook is a web-based environment that allows users to comfortably combine text, code (including execution and results), pictures, etc. in one file. A front-end part is accessible via an internet browser, a back-end part runs either on a local machine or on a remote server.

Although Python can be comfortable for implementing and testing trading strategies, there are programming languages that are faster. When choosing the appropriate programming language, a user's intentions will determine the level of performance and therefore the specific programming language needed. It is possible to improve Python's performance, for example with the use of *Cython*[7] or *PyPy*[8].

3.3 Backtesting

To find out if the trading strategy is successful or not, it is appropriate and recommended to research it using historical data. This process is called *backtesting*. There are some frameworks in Python that allow backtesting in addition to providing other features such as live trading or some other interaction with brokers. Let us do a brief research on the well-known frameworks in Python. All of them support backtesting and live trading.

[1] The pandas development team. *pandas-dev/pandas: Pandas* [online]. [Cit. 15.2.2023]. Available at: `https://doi.org/10.5281/zenodo.3509134`

[2] *SciPy* [online]. [Cit. 15.2.2023]. Available at: `https://scipy.org/`

[3] Seabold, Skipper and Perktold, Josef. *Statsmodels: Econometric and statistical modeling with python* [online]. [Cit. 15.2.2023]. Available at: `https://www.statsmodels.org`

[4] Hunter, J. D. *Matplotlib: A 2D graphics environment* [online]. IEEE COMPUTER SOC, 2007. [Cit. 15.2.2023]. Available at: `https://matplotlib.org/`

[5] Michael L. Waskom. *Seaborn: statistical data visualization* [online]. The Open Journal, 2021. [Cit. 15.2.2023]. Available at: `https://doi.org/10.21105/joss.03021`

[6] *jupyter* [online]. [Cit. 15.2.2023]. Available at: `https://jupyter.org/`

[7] *Cython: C-Extensions for Python* [online]. [Cit. 15.2.2023]. Available at: `https://cython.org/`

[8] *PyPy* [online]. [Cit. 15.2.2023]. Available at: `https://www.pypy.org/`

- *Quantopian* was a leading platform in the field with the aim to create a crowd-sourced hedge fund. It existed from the year 2011 to the year 2020. It has created two trading tools in Python called *zipline* and *zipline-live*. The support and development of the tools ended with the end of the project in 2020, when shut down due to low efficiency of its investment strategies. The Quantopian community has divided itself. The big part went over to the main competitor QuantConnect or to free projects such as Freqtrade, or Backtrader. Some members created a fork of zipline, but it is unclear if it is still active. Lastly, there are a lot of small projects which are in some way connected with zipline.[9,10]

- *QuantConnect* is an open-source platform for algorithmic trading based on cloud service. It was founded in the year 2011. QuantConnect offers its users a work environment in form of its website or its desktop application *LEAN (Lean Algorithmic Trading Engine)*. As QuantConnect is mainly paid service it is wealthy in its features. From the support of various classical brokerages and cryptocurrency exchanges to tools for team collaboration for institutional customers. The free version is cut only to provide fundamental data and backtesting.[11]

- *Freqtrade* is a free, open-source crypto trading bot. It has a large and active community and active project. It supports all major exchanges. The uncommon thing and also an advantage is that it contains strategy optimization by machine learning.[12]

- *Backtrader* is free, easy to use, has a large and active community, good documentation, and live trading. The disadvantages are low project activity and that it does not support cryptocurrency marketplaces.[13]

Besides those mentioned, there are plenty of other backtesting frameworks in Python, both bigger (for example *CCXT*[14]), and smaller. Smaller frameworks usually have a smaller community of users and resources. However, the tools offered are often similar to the previously mentioned frameworks and it is also possible to find

[9] *zipline-live* [online]. [Cit. 15.2.2023]. Available at: `http://www.zipline-live.io`

[10] Wikipedia contributors. Cryptocurrency. *Wikipedia, The Free Encyclopedia* [online]. Last updated 5.2.2023. [Cit. 15.2.2023]. Available at: `https://en.wikipedia.org/wiki/Quantopian`

[11] *QuantConnect* [online]. [Cit. 15.2.2023]. Available at: `https://www.quantconnect.com`

[12] *Freqtrade* [online]. [Cit. 15.2.2023]. Available at: `https://www.freqtrade.io`

[13] *Backtrader* [online]. [Cit. 15.2.2023]. Available at: `https://www.backtrader.com`

[14] *ccxt* [online]. [Cit. 16.2.2023]. Available at: `https://github.com/ccxt/ccxt`

Table 3.1: Statistical properties of interarrival times in ms

	Min	Max	Mean	Median
Tick	51	70482	496	153
Binance	0	7892	205	50

some advantages. Examples could be *vectorbt*[15] or *Backtesting.py*[16]. In the thesis, we use the Python framework *backtrader* to backtest strategies. We chose backtrader because there is a good relation between the features offered and difficulty of use. Above that, it has a large, active community.

3.4 Statistical distribution of data

This section is focused on the statistical distribution of *Interarrival Times* (IaT) of the data. IaT are differences between times of single observations. In the thesis, IaT is represented by the amount of time that passes between two quotes (Tick data) or two trades (Binance data). The description of how we proceeded in the search for distribution is described below.

3.4.1 Data adjustment for distribution fitting

Cartea; Jaimungal; Penalva, 2015, p. 48 states that in their data, "33 % of the time there are no price changes". Therefore, they only consider instances where there is a price change in ask or bid. In Tick data, there are just 237 cases from total 174,486 where there is no price change, whereas in Binance data it is 463,631 cases from 885,623. In both cases, we removed the quotes/trades where there was no price change. If a trade had the same price as a previous trade, it was removed. After this process, the IaT are calculated.

As we can see in Table 3.1, there is a significant difference between the minimum and maximum of IaT. Cartea; Jaimungal; Penalva, 2015, p. 48 deals with heavy tails when fitting distribution to their times. For these reasons, we will work with whole datasets and with cut datasets where the IaT for Tick data and Binance data are under 2000 ms and 935 ms, respectively. Both thresholds approximately correspond to the 95th percentile.

[15] *vectorbt* [online]. [Cit. 16.2.2023]. Available at: `https://vectorbt.dev/`

[16] *Backtesting.py* [online]. [Cit. 16.2.2023]. Available at: `https://kernc.github.io/backtesting.py/`

3.4.2 Fitting distributions to interarrival times

The Python packages `fitter`[17] and `distfit`[18] provide an automatic statistical distribution fitting with the utilization of up to 80 and 89 distributions respectively, from package `Scipy`. With the same setting, one gets the same result from both packages. In the case of the package `fitter`, the best distribution is chosen by default by an error sum of squares method (SSE), in the case of the package `distfit` the types of methods can be: parametric (methods SSE also known as RSS, Wasserstein distance, Kolmogorov-Smirnov test, Energy distance), non-parametric (the quantile and percentile method), and discrete. The parametric methods for the goodness of fit tests impact just the selection of the best-fitted distribution, the estimated parameters are always the same.

Besides the goodness of fit test, we plot Q-Q plots of best-fitted distributions to check the results visually.

3.4.2.1 Bin size

Both packages offer the possibility to specify the number of bins. The number of bins has an impact on a histogram plot, but also on the selection of the best-fitted distribution. With the various number of bins, the packages can return various values of sumsquare error. Therefore, if we do a fit with the number of bins of 50 the best-fitted distribution can be different than with the number of 100. However, the estimated parameters of distributions are always the same.

For these reasons, we use *Freedman–Diaconis* rule to determine the bin size. The formula of the rule is:

$$h = \frac{2 \times \mathrm{IQR}}{\sqrt[3]{n}}, \tag{3.1}$$

where h is bin size, IQR is the interquartile range, and n is the number of observations. The interquartile range is the difference between the third quartile and the first quartile. Then we can count the number of bins b for IaT series $Y = \{y_1, \ldots, y_n\}$[19]:

$$b = \frac{\max\{y_1, \ldots, y_n\} - \min\{y_1, \ldots, y_n\}}{h}. \tag{3.2}$$

[17] *fitter* [online]. [Cit. 5.3.2023]. Available at: `https://fitter.readthedocs.io`

[18] *distfit* [online]. [Cit. 5.3.2023]. Available at: `https://fitter.readthedocs.io/en/latest/`

[19] numpy.histogram_bin_edges. *NumPy* [online]. [Cit. 5.3.2023]. Available at: `https://numpy.org/devdocs/reference/generated/numpy.histogram_bin_edges.html`

3.4.3 Fitting distributions over various time frames of a day

In the book Cartea; Jaimungal; Penalva, 2015, p. 49, Cartea et al. found out that their IaT, using the 95th percentile, has the power law distribution. With worse results, they also tried to fit the exponential distribution. If there is evidence that our IaT have an exponential distribution, we could use continuous Hidden Markov models for forecasting. Therefore, we tried to fit the power law and the exponential distributions over various time frames of IaT within a day.

As in Cartea; Jaimungal; Penalva, 2015, we work with the 95th percentile of IaT and in addition with the whole dataset. Tested time frames are 5, 10, 30, and 60 minutes long. Taking an example of a time frame of 30 minutes, the algorithm tests 48 time frames per day. When starting at midnight, the first time frames look as 00:00–00:30, 00:30–01:00, …. To check if the IaT of a time frame follows the power law or exponential distribution we utilize the one-sample Kolmogorov-Smirnov test.

With a time frame of 30 minutes, for example, the first frame begins at midnight and the rest of the day splits into 30-minute segments (e.g. 00:00–00:30, 00:30–01:00, etc.) for a total of 48 time frames per day.

3.4.4 Test if the interarrival times follow the same distribution during a day

In addition to searching for a distribution of a time frame of a day, we also tested if the IaT has the same distribution in the two following time frames. An algorithm is similar to the one described in the subsection above 3.4.3. The time frames are 5, 10, 30, and 60 minutes long. An example with a 30-minute time frame, the algorithm tests the time frame 00:30–01:00 with 00:00–00:30, 01:00–01:30 with 00:30–01:00, etc. Whether 2 time frames follow the same distribution is tested with two-sample Kolmogorov-Smirnov test. Again, we work with the 95th percentile of IaT and with the whole dataset.

3.5 Trading strategies

The trading strategy is a strictly given plan for buying and selling an asset. The main strategy used in this thesis is based on the forecast of the ARIMA model. In order to be able to compare to have a possibility to compare the strategy with some strategies not based on statistical model, we chose two basic trading strategies: mean-reversion and momentum strategy. They are strategies that utilize particularly moving averages.

Notes:

1. We look at data as on series of values, not times. Thus when we talk about the length of a moving average, the unit is a number of values. The same with forecasts.

2. For simplicity, we work only with speculations on a price increase (long position).

3.5.1 Mean-reversion strategy

Mean-reversion strategies work on the assumption that a price of an asset returns to its mean, i.e. if the price is above the mean, after some time it goes down to the mean, and if the price is below, it goes up to the mean. There are various ways of implementation, so one could say the term mean-reversion strategy refers to a group of strategies, not to one specific. Inspiration for the strategy comes from this link[20].

3.5.1.1 Implementation

The strategy is based on the distance of the price from the mean. To measure the distance of the price from the mean, we use the so-called z-score. It tells us how many standard deviations the price is above or below the mean. The formula of the z-score is:

$$z = \frac{x - \mu}{\sigma}, \tag{3.3}$$

where x is the price, μ is the mean and σ is the standard deviation.

If $z < -1$, i.e. if the price is one or more standard deviations below the mean, it is a signal to buy. If we had also traded short positions in the thesis, we would open the position when $z > 1$. If $|z| < 0.5$ it is a signal to sell, because the next move of the price is less certain.

There are two options for the type of mean. First is the *cumulative moving average* (CMA). It is a moving average with a fixed start position. We set a fixed start from when the mean will be calculated. For example, the fixed position is the beginning of a day, then it continues as a classic moving average - each incoming value is added to the sum and the number of values is increased by one. The second option is a classic simple moving average. The size of the rolling window is a variable to optimize.

[20] Mean reversion Models. *Github* [online]. [Cit. 10.3.2023]. Available at: `https://github.com/Auquan/Tutorials/blob/master/Mean%20reversion.ipynb`

3.5.2 Momentum strategy

In contrast with mean-reversion strategies, momentum strategies proceed from the presumption that the price which is going up will continue going up and the price which is going down will continue going down. As in the case of the mean-reversion strategy, there is not one uniform form of implementation. Inspiration for the strategy comes from these links[21] and[22].

3.5.2.1 Implementation

The chosen implementation is called moving averages ribbons. It uses a set of simple moving averages with different lengths. The buy signal is generated when the moving averages are in increasing order (example):

$$\text{MA}(10) < \text{MA}(20) < \ldots < \text{MA}(90) < \text{MA}(100)$$

and in the opposite way the sell signal:

$$\text{MA}(10) > \text{MA}(20) > \ldots > \text{MA}(90) > \text{MA}(100)$$

In the parentheses is the length of the moving average. The number of moving averages and their length is a subject of optimization.

The visualization of a relation between the single moving averages we can see on the graph in Figure 3.1. The graph is generated on the foundation of data obtained during the backtesting of this strategy. Buy order (MA(10) < MA(20) < …) is created when the 1261 trade passed and sell order (MA(10) > MA(20) > …) when the 1463 trade passed.

3.5.3 ARIMA strategy

An ARIMA strategy exploits a forecast from an ARIMA model. In the case of the thesis, the data on which the forecasts are made are logarithmic returns of prices (see 2.6.7). Thus, a change in a price is forecasted, not the price itself. Inspiration for the strategy comes from the website[23] and from Halls-Moore, 2016, p. 369.

[21] Momentum Strategies. *Github* [online]. [Cit. 10.3.2023]. Available at: `https://github.com/Auquan/Tutorials/blob/master/Momentum%20Strategies.ipynb`

[22] Cory Mitchell. Moving Average Ribbon: Definition, Meaning, Calculation Formula. *Investopedia* [online]. Last updated 30.6.2022. [Cit. 10.3.2023]. Available at: `https://www.investopedia.com/terms/m/movingaverageribbon.asp`

[23] ARIMA+GARCH Trading Strategy on the S&P500 Stock. *Github* [online]. [Cit. 10.3.2023]. Available at: `https://github.com/Auquan/Tutorials/blob/master/ARIMA%20%2B%20GARCH%20to%20model%20SPX%20returns.ipynb`

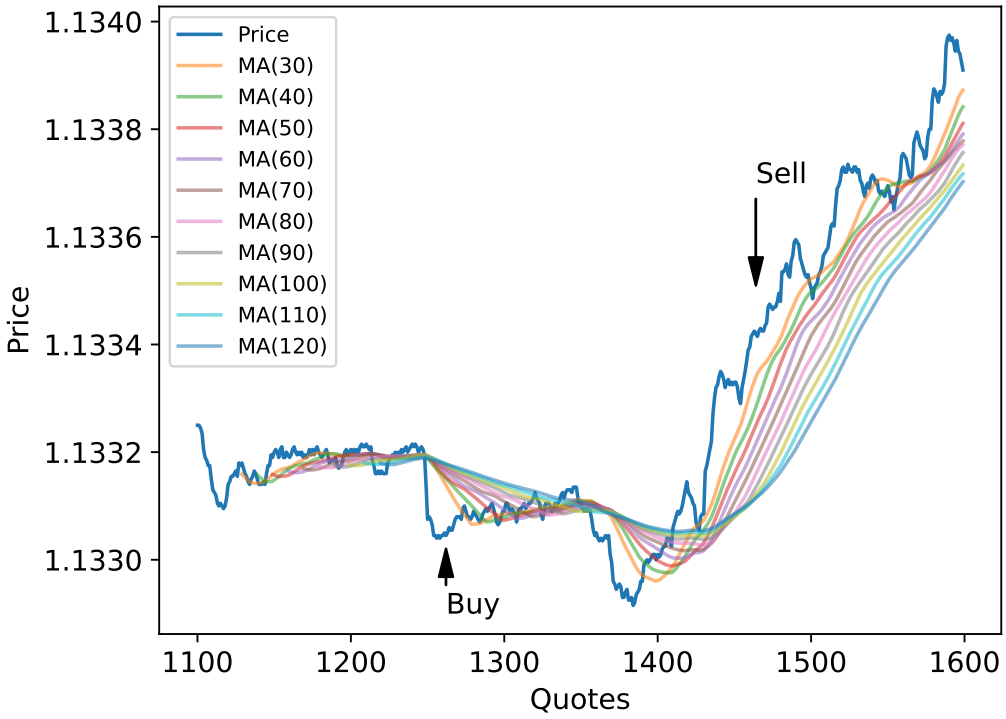

Figure 3.1: Moving average ribbons

3.5.3.1 Implementation

In order to use an ARIMA strategy, it is necessary to fit an ARIMA model. The process of fitting the ARIMA model to the data is described in section 3.6. The mathematical definition of the ARIMA model is in section 2.5. Following is a general description of the steps of the ARIMA strategy.

1. Determine an order of a model.
2. Choose the length of a forecast.
3. Fit the model to data.
4. Make the forecast.
5. Trade according to the information from the forecast.

The length of the forecast is subject to optimization.

Generally, a buy signal is generated when the forecast is greater than 0, i.e. the next price should be greater than the current price according to the forecast. Then the algorithm waits till the forecast is less than 0, i.e. the next price should be less than the current price, and the sell signal is generated.

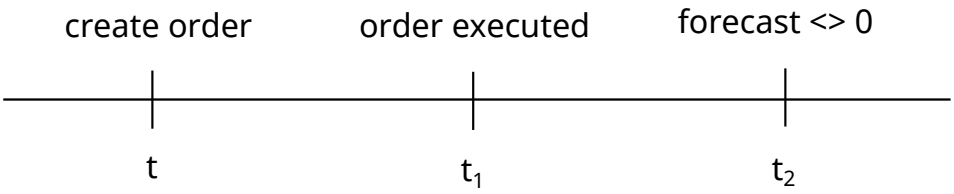

Figure 3.2: Forecast timeline

The implementation in a code is more complicated due to shifts between information from the forecast, creating a trade order, and executing the order. It is shown in Figure 3.2. The forecast expresses the change between prices at times t_2 and t_1. So we want to execute the order at t_1 because the price is lesser/greater than at t_2. Thus we must create the order at time t and it will be executed for the price at t_1.

3.6 ARIMA fitting

The process of ARIMA fitting is as follows:

1. Determine the number of differencing. The series must be stationary before the determination of AR and MA terms.

2. Determine the number of AR and MA terms.

3. Fit the model and check if a good fit was achieved, else edit the order. The residuals of the fitted model must be white noise.

There are two options for determining the order of the model; the manual option which relies on visual checks by a person, and the automatic option which relies on the use of software methods. From our experience, it is better to combine software methods with visual checks.

3.6.1 Manual order determination

Manual fitting is based on considering plots of an ACF and PACF of residuals of the fitted model. After every change in the order of the model, followed by fitting the model, we must check the plot and according to suggestions from the plot consider how to continue and what terms to add or remove. There are a few recommendations for each of the ARIMA processes below, including a Figure 3.3 with instructions.

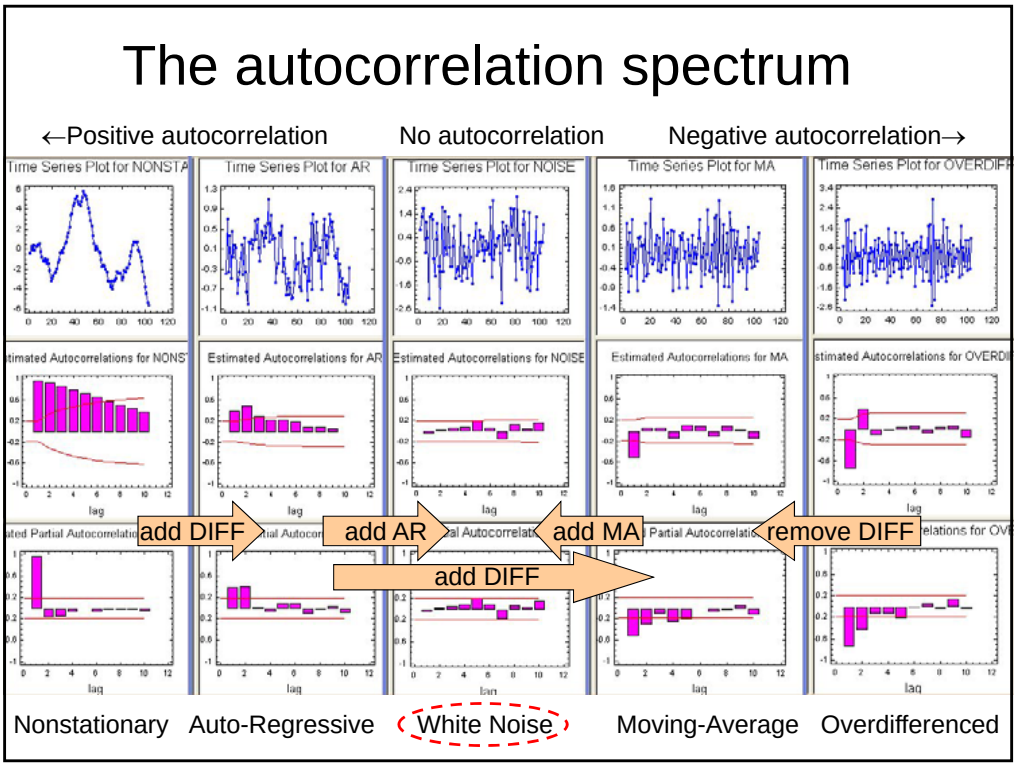

Figure 3.3: Order determination instructions, Nau, 2020

3.6.1.1 Determination of the order of differencing

- If residuals have positive, significant autocorrelation to a high number of lags, we should consider increasing the order of differencing.
- If residuals have small autocorrelation or if the lag 1 is zero or negative, it is a suggestion that there is no need for a higher number of differencing. Also, when the lag 1 is -0.5 or smaller it is a sign of overdifferencing.
- The best order of differencing is usually the one when the series has the lowest standard deviation.

3.6.1.2 Determination of the order of AR terms

- If ACF decays slowly and PACF has a sharp cut-off, we should consider adding an AR term. The number of AR terms is indicated by the lag of the PACF cut-off.
- If ACF seems underdifferenced, we should consider adding an AR term.

- If the sum of AR parameters is almost 1 (there is a unit root in AR), we should try to reduce the order of AR and increase differencing.

3.6.1.3 Determination of the order of MA terms

The recommendations are the opposite of those in the AR case.

- If ACF has a sharp cut-off and PACF decays slowly, we should consider adding a MA term. The number of MA terms is indicated by the lag ACF cut-off.
- If ACF seems overdifferenced, we should consider adding an MA term.
- If the sum of MA parameters is almost 1 (there is a unit root in MA), we should try to reduce the order of MA and reduce differencing.

3.6.2 Automatic order determination

Another option for identifying the order of the ARIMA model is the use of software methods. The automatic order determination utilizes principles other than those mentioned in subsection 3.6.1.

We briefly introduce two functions in Python that both allow determining the whole order completely automatically.

- The function called `x13_arima_select_order`[24] in the package `statsmodels`.
- The function `auto_arima` in the package `pmdarima`[25]. The package `pmdarima` wraps the package `statsmodels` and it is the equivalent to `auto.arima` functionality in the programming language *R*. It offers the order determination of nonseasonal and seasonal ARIMA model. One can choose from various statistical tests to slightly affect the process and from various information criteria for comparing the orders between each other. Besides classical "brute force" determination, when each combination of AR, MA, ... terms is tried, it offers Hyndman and Khandakar algorithm which is more sophisticated and thus faster.

When one does not want to use the functions mentioned above, one can write the code for order determination by themself. The basic logic for the code is described in the subsections below.

[24] Seabold, Skipper and Perktold, Josef. statsmodels.tsa.x13.x13_arima_select_order.*statsmodels: Econometric and statistical modeling with python* [online]. [Cit. 30.3.2023]. Available at: `https://www.statsmodels.org/dev/generated/statsmodels.tsa.x13.x13_arima_select_order.html`

[25] pmdarima. PyPi [online]. Last updated 15.3.2023. [Cit. 30.3.2023]. Available at: `https://pypi.org/project/pmdarima/`

3.6.2.1 Determination of the order of differencing

The best solution is to determine the order of differencing manually, as it is described in 3.6.1.1. The model selection with the use of the software is based on comparing an AIC of a fitted model. If we compare fitted models with different differencing by AIC, we get biased results.

Also, we can use the Augmented Dickey-Fuller test to check if the series is stationary. Rejecting the hypothesis H_0 is in favor of the series's stationarity.

3.6.2.2 Determination of the order of AR and MA

When we have the order of differencing, we can determine the remaining: AR and MA terms. In a nested for loop we try to fit various numbers of AR and MA terms. Then we choose the fitted model with the lowest AIC.

3.6.3 Check

After fitting the model, in both manual and automatic cases, we need to check the residuals of the fitted model. We want the residuals to be white noise. As a test, if residuals are not autocorrelated, we utilize the Ljung-Box test. If the test shows that the residuals are autocorrelated, this signifies that we chose the wrong order of the model or even the wrong type of statistical model.

Results 4

4.1 Data

In this thesis, we work with two datasets from different sources and of different natures. The first is historical tick data on the currency pair EUR/USD, the second is historical tick data on the cryptocurrency pair BTC/USDT. The difference between the two datasets is described in subsection 4.1.3.

4.1.1 Dataset EUR/USD

The source of the data is the website *exacttrading.com*[1]. It offers datasets of several currency pairs from years 2003 to 2020 for free download. According to the information provided by Paul Langham, the person who stands behind the website, the data comes from Dukascopy Bank and "gives every up or down quote for the instrument being measured". Dukascopy Bank is a Swiss bank that besides banking and other financial services provides online trading services with a focus on foreign exchange, bullion, CFD, and binaries. [2]

As we said at the beginning of this section we will work on pair EUR/USD. The newest dataset contains data from the beginning of the year 2014 to the beginning of the year 2020. The file with the dataset has 159 million rows and its size is 7.1 GB. Working with a file of this size is highly demanding on computational resources. Therefore, we chose one reference day for effective work. The day is 3 January 2019 and it is the day with the most quotes in the year 2019, specifically 174,487. The visualization of data from 3 January 2019 is in Figure 4.1.

The downloaded dataset contains these columns separated by commas: datetime with millisecond accuracy, ask price, bid price, and two empty columns. The file does not have a header. Before the work, we prepared the data. We did data cleaning and

[1]Tick data. *Exact Trading* [online]. [Cit. 15.3.2023]. Available at: `https://exacttrading.com/historical-fx-tick-data-2/`

[2]Company. *Dukascopy* [online]. [Cit. 15.3.2023]. Available at: `https://www.dukascopy.com/swiss/english/about/company/`

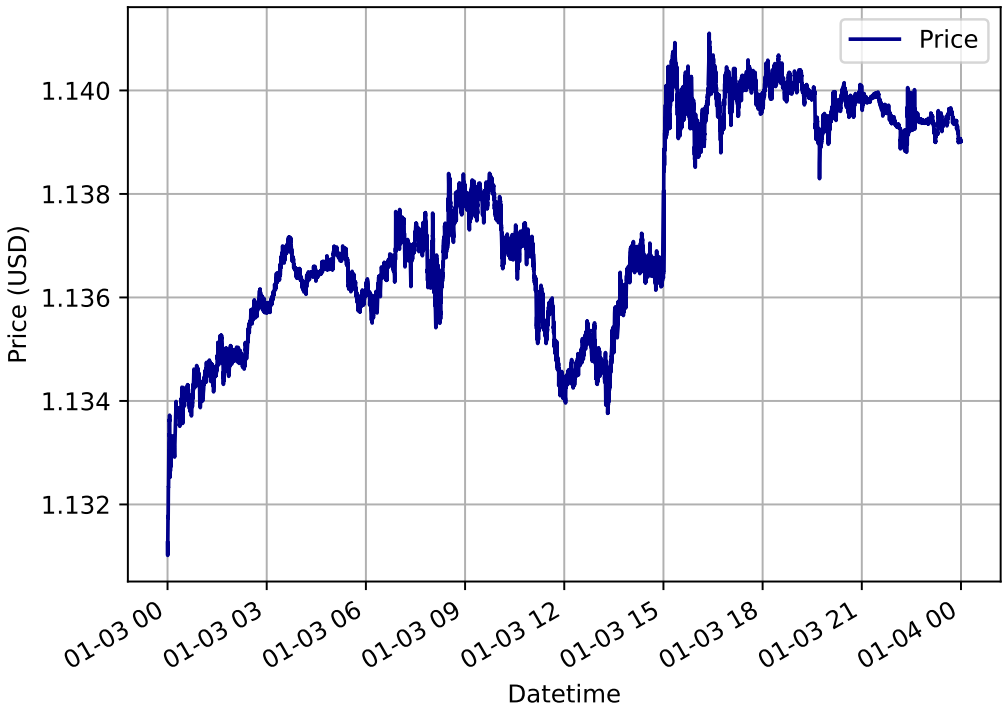

Figure 4.1: Price of Tick data, 3 January 2019

some changes for convenience work. We replaced NaN values with zeros, created header and indexing (from zero), dropped the two empty columns, and changed datetime format to ISO 8601 format (YYYY-MM-DD hh:mm:ss.sss). This data from exacttrading.com will be referred to as "Tick data" in this thesis.

4.1.1.1 Data check

As a check whether the data can be considered relevant, we compare it with data from *Yahoo!*. As Yahoo! data is daily-based and OHLC typed (open, high, low, close) we transformed the tick data into the same format. Furthermore, it is necessary to merge it with Yahoo! data as a key, because Yahoo! data contains fewer days than transformed tick data.

We can see on a graph in Figure 4.2 the differences in close prices of Tick and Yahoo! data. It is worth mentioning that high or low prices are often different among exchanges.

4.1.1.2 Histogram

In order to have an idea of how the year 2019 stands in comparison to other years in the number of quotes, we will take a look at the evolution of the number of quotes

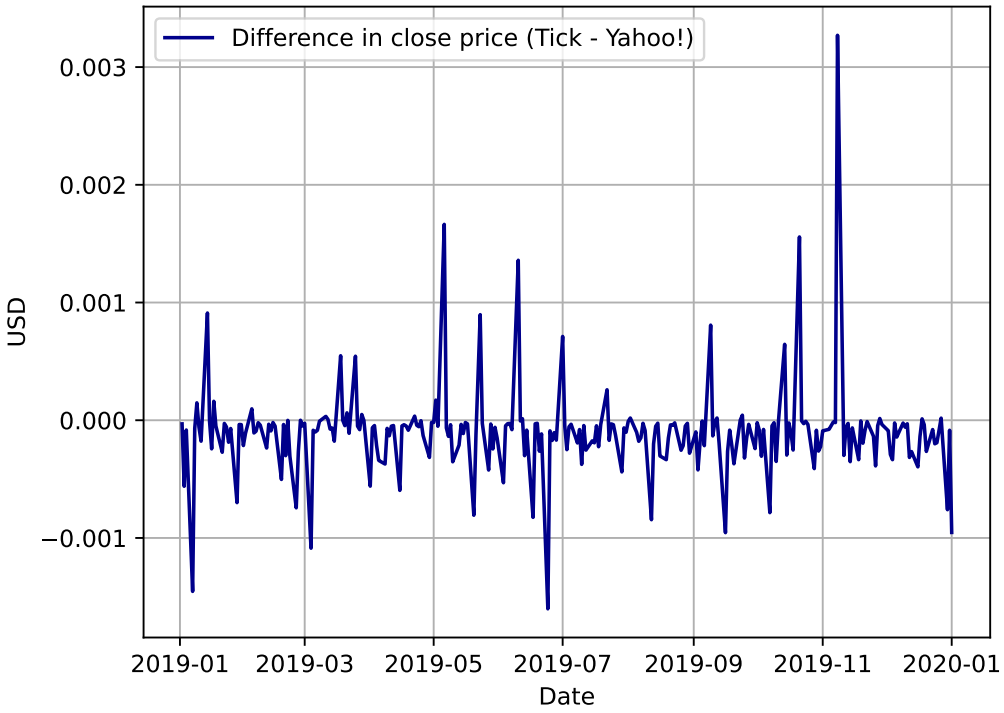

Figure 4.2: The difference between Tick and Yahoo! data in close price

in time.

On the histogram graph in Figure 4.3 we can see the number of quotes per years 2004-2019. Significant (different from others) are the years 2008-2010 and 2016. When comparing years, except for the fluctuation years, the mean of the number of quotes for the first 4 years is 13.1 million, for 2011-2019 (except 2016) is 22.7 million. That is an increase of 73 %. Taking only the years 2011-2019 (except 2016), the year 2019, which has 25 million quotes, is above the mean, but not significantly as in the case of the year 2016. We can say that it is an ordinary year.

4.1.2 Dataset BTC/USDT

The data comes from the cryptocurrency exchange *Binance*. Specifically, it is downloaded from the webpage[3]. Binance is the largest cryptocurrency exchange in the world in the terms of the trading volume. Regarding data, it provides several types of historical data from spot trading and from futures trading. Spot data offers: candlestick data, tick data, and aggregate trades data. In this thesis, we work with tick data. The data are free to download.

[3]Historical Market Data. *Binance* [online]. [Cit. 15.3.2023]. Available at: `https://www.binance.com/en/landing/data`

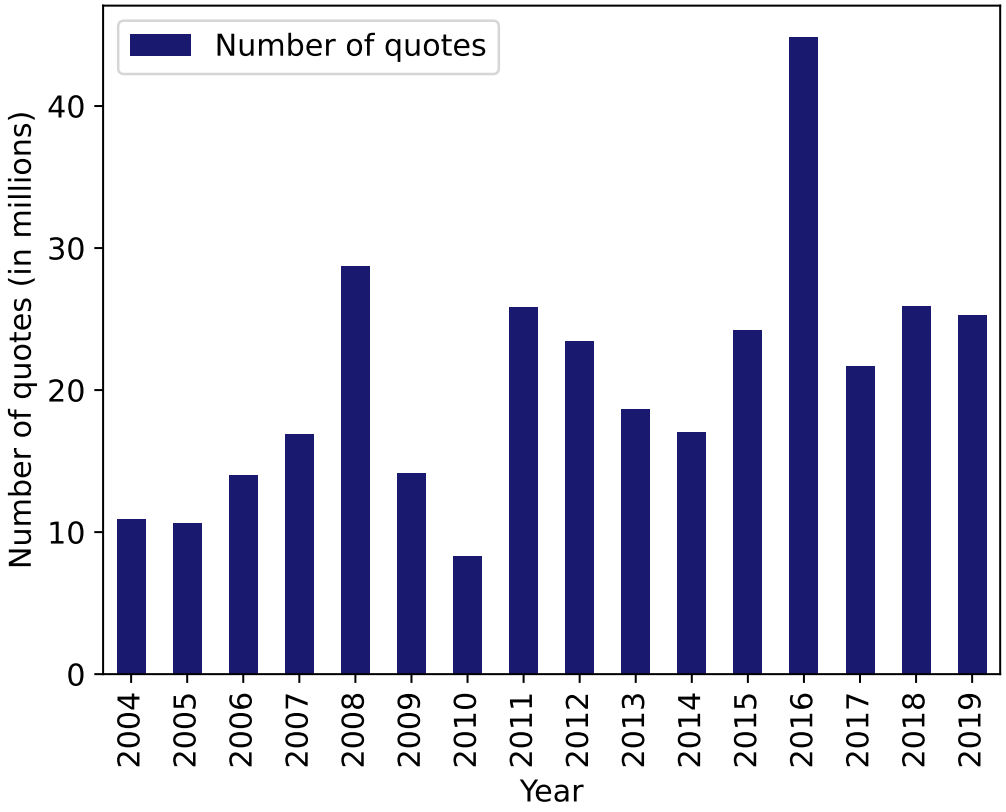

Figure 4.3: Histogram with number of quotes per year

Tick data from Binance are executed trades. It contains columns: `tradeID`, `price`, `qty`, `quoteQty`, `time`, `isBuyerMaker`, `isBestMatch`. Columns `tradeID`, `price`, and `time` are self-explanatory. The column `qty` is a quantity (in our case of Bitcoin), and `quoteQty` also means quantity (in our case of USDT). There is no information available about column `isBestMatch` . Column `isBuyerMaker` is more complicated; it includes information on whether the quote of the trade was in the order book or not and whether the trade was buy or sell (regarding one of the two assets). If there is a value `True` in that column it means that a person who wanted to buy Bitcoin created a buy order. The buy order was not executed immediately, but at first, it was waiting in the order book and then was fulfilled with the sell order of another person. The buyer is a maker, the seller is a taker. A value `False` means the opposite. A person who wanted to sell Bitcoin created a sell order. After some time of waiting in the order book the sell order was fulfilled with the buy order of another person. The seller is a maker, the buyer is a taker.

Regarding data adjustment, there were fewer changes than in Tick data. We replaced NaN values, added our own indexing, and also added a column with transformed datetime. The `time` column is in the format of Unix time, we transformed

Figure 4.4: Price of Binance data, 3 January 2022

it to ISO 8601 format (YYYY-MM-DD hh:mm:ss.sss).

For Tick data, we work only with one day. In order to have data similar we chose the day 3 January 2022 for work with Binance data. From the columns are important for us the column with prices and with times of trades. The data from Binance will be referred to as "Binance data" in this thesis. The visualization of the data we can see in Figure 4.4.

4.1.3 Tick vs Binance data

In the subsection 2.8 we write about tick data classification. In that sense, we have not classified our data yet. The tick data is case two – it is containing information about best bid/ask quotes. It is the top of the order book, there is no information about the trades. The Binance data is case number one – it is containing information about each executed trade. There is no information about bid/ask prices.

4.2 Distribution fitting

In this section, there are results of distribution fitting to interarrival times of data. The methodology is described in section 3.4.

Table 4.1: Results of fitting distributions to the interarrival times

	Whole data	Threshold
Tick	Student's t-distribution	Double Weibull distribution
Binance	Generalized extreme value distribution	Log-Laplace distribution

4.2.0.1 Fitting distributions to day data

Results of fitting distributions to the IaT of one whole day are in Table 4.1. On Q-Q plots in Figure 4.5 we can see that the IaT does not follow any of the distributions which the software chose as the best.

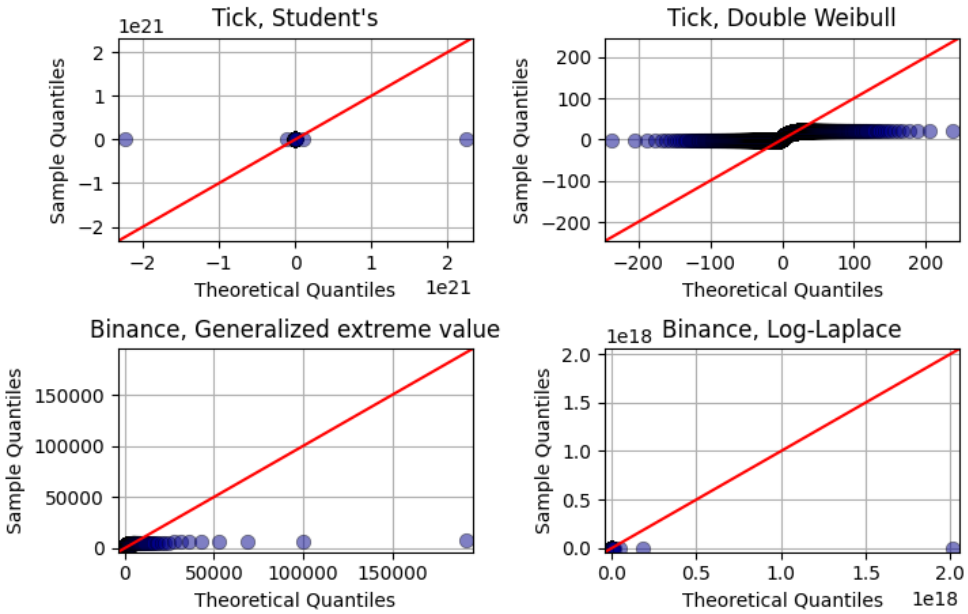

Figure 4.5: Q-Q plots of the best-fitted distributions

4.2.0.2 Fitting distributions over various time frames of a day

Tick and Binance IaT do not follow exponential or power law distribution on any of mentioned time frames. H_0 hypotheses of the Kolmogorov-Smirnov test are rejected in all cases.

4.2.0.3 Test if the interarrival times follow the same distribution during a day

The IaT do not follow the same distribution during the day. We can see the exact results in Table 4.2. There is a number of rejected H_0 hypotheses related to the total

Table 4.2: Test on the same distribution, number of rejected H_0 hypothesis

	Time frame	Whole data	Threshold
Tick	5 min	89/287	59/287
	10 min	67/143	47/143
	30 min	35/47	30/47
	60 min	21/23	19/23
Binance	5 min	208/287	203/287
	10 min	126/143	123/143
	30 min	46/47	45/47
	60 min	21/23	22/23

number of cases.

4.2.1 Conclusion

We did not find any distribution which would fit the interarrival times of the data. Therefore, we cannot use the previously mentioned continuous Hidden Markov model to make forecasts for series. We will look at the series in terms of changes in price, not in terms of changes in time.

4.3 ARIMA fitting

We try to determine an order for the ARIMA model. As an example of the ARIMA fitting process, we try to fit ARIMA to Tick data for one day (3 February 2019). This part is in the subsection 4.3.1. Complex fitting results are in the subsection 4.3.2.

4.3.1 Example

Below is an example of the process of fitting to a day Tick data.

1. At first, we do logarithmic returns. It is important to note, that during the process of making logarithmic returns, the series is one time differenced. Therefore the series will be likely already stationary.

2. We identify the order of differencing. On graph in Figure 4.6 we can see the ACF and PACF function of logarithmic returns. The behavior of the autocorrelations follows the second point in the subsection 3.6.1.1, above that, Augmented Dickey-Fuller test rejected the H_0 on confidence interval 95%. We can say that the series is stationary. To be sure, in Figure 4.7 we can see that behavior of the first difference exhibits signs of an overdifferencing (see 3.6.1.1).

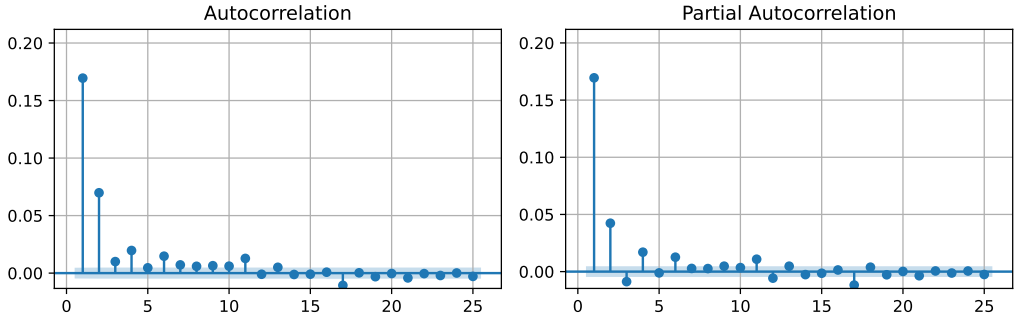

Figure 4.6: ACF, PACF - logarithmic returns

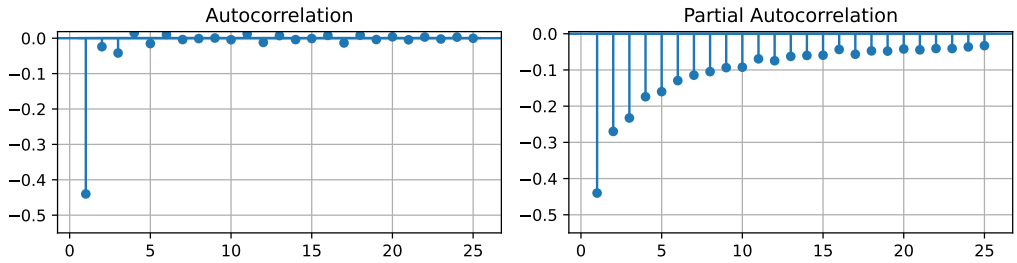

Figure 4.7: ACF, PACF - ARIMA(0, 1, 0)

3. Considering the plot of logarithmic returns in Figure 4.6 and following the notes from the subsections 3.6.1.2 and 3.6.1.3 we consider the ARIMA order $(2, 0, 2)$ as the most appropriate. Nevertheless, on the ACF and PACF plot in Figure 4.8 we can see, that the fitted model do not capture autocorellation at higher lags. The p-value of Ljung-Box test is 4.43×10^{-13} and we reject the H_0 hypothesis on confidence level 95 %. The residuals of the fitted model exhibit serial correlation. The ARIMA$(2, 0, 2)$ is not a good fit.

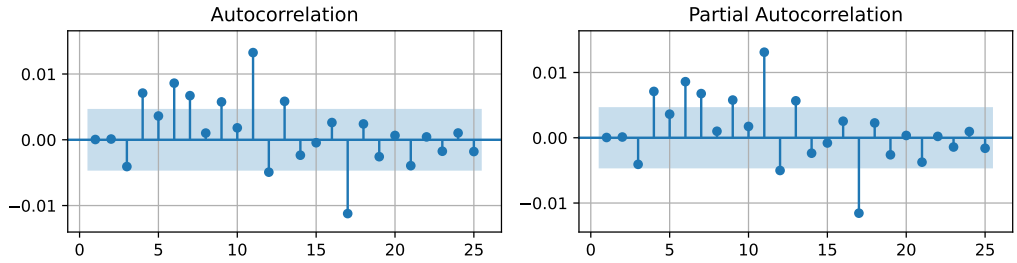

Figure 4.8: ACF, PACF - ARIMA(2, 0, 2)

4. The package `pmdarima` determined the order as $(3, 0, 2)$. The AIC of ARIMA$(2, 0, 2)$ is $-3{,}482{,}749.999$ and ARIMA$(3, 0, 2)$ is $-3{,}482{,}789.591$. Thus, ARIMA$(3, 0, 2)$

should be a better model. The ACF and PACF plot in Figure 4.9 looks better than in the case of ARIMA(2, 0, 2), however, we can see that neither (3, 0, 2) order does capture autocorrelation at higher lags. The p-value of Ljung-Box test is 0.000004, thus we have evidence to reject the H_0 hypothesis on 95% confidence interval. Neither ARIMA(3, 0, 2) is a good fit.

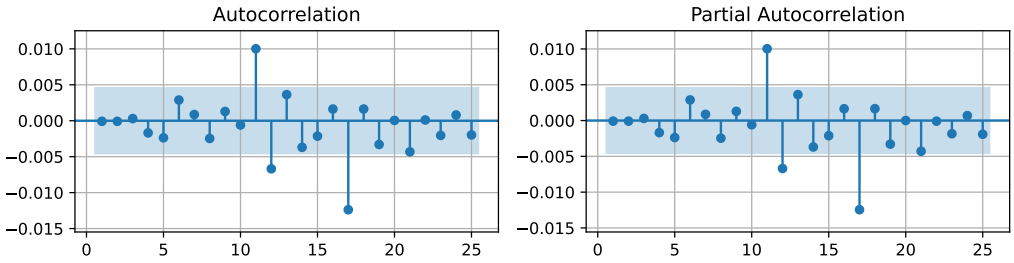

Figure 4.9: ACF, PACF - ARIMA(3, 0, 2)

Table 4.3: ARIMA fitting results – Tick data

Time frame	ARIMA order	Number of orders	Number of valid orders
30 min	(1, 0, 0)	4	4
30 min	(1, 0, 2)	1	1
30 min	(0, 0, 3)	1	1
30 min	(0, 0, 2)	2	2
30 min	(2, 0, 0)	1	1
30 min	(3, 0, 0)	1	0
60 min	(0, 0, 2)	2	2
60 min	(3, 0, 0)	1	0
60 min	(3, 1, 0)	1	0
60 min	(1, 0, 0)	3	1
60 min	(2, 0, 0)	1	0
60 min	(1, 0, 2)	1	1
60 min	(2, 0, 1)	1	1
12 hours	(0, 0, 2)	3	0
12 hours	(2, 0, 1)	7	0
1 day	(3, 0, 2)	1	0

4.3.2 Complex results

We tried to fit an ARIMA model to time frames of various length during a day. The results for Tick data are in Table 4.3, for Binance data in Table 4.4. The search algorithm is described in the paragraph below.

We chose four time frames: 30 minutes, 60 minutes, 12 hours, and 1 day. For all time frames excluding the 1 day time frame, we took ten random data parts of a time frame's length and tried to determine the order of the ARIMA model. In the results tables we can see that for various data parts of the same length, there are various determined orders. The column `Number of orders` expresses how many times the specific order was determined for a data part. The determined order was not a good fit in every instance, therefore the column `Number of valid orders` gives us information on how many determined orders are considered as a good fit (based on a Ljung-Box test).

In both Tick and Binance results, there are many different orders for various data parts of the same length. The determined orders for Binance data were not a good fit in 26/31 total cases, while the determined orders for Tick data were not a good fit in 17/31 cases. However, for Tick results, the determined orders in the 30-minute time frame were overwhelmingly a good fit. The best result overall is the first row in Table 4.3. For 4 from 10 data parts the same order (1,0,0) was determined and each one is a good fit. Generally, these results mean that if we successfully determine the order of the model and then fit the model, it will most likely not be valid after a short period of time.

To conclude, the ARIMA model is not a proper statistical model for our data. In the following we compare the strategy exploiting a badly chosen but sophisticated statistical model with the basic trading strategies.

4.4 Trading strategies

In this section, we discuss the results of the strategies described in section 3.5. When testing trading strategies we try various parameters, however, we do not do an exhaustive optimization. Our tested parameters can be used as a sign of direction for further optimization.

As we noted in the subsection 3.5, we look at data in terms of values, not times. However, working with a window, for example, of 5000 trades does not provide any information about the length of the window in the sense of time. Therefore we counted the average number of values per some basic time frames. The results are in Table 4.5. Now we know, that when we work with e.g. frame of 605 quotes of Tick data, it approximately equals 5 minutes.

In all results tables of trading strategies are common columns: `O`, `T+`, `T-`, `T=`, `Profit (+) / Loss (-)`. The column `O` is the number of executed orders, `T+` is the number of profitable trades, `T-` is the number of loss-making trades, `T=` is the number of trades where there was no profit or loss, `Profit (+) / Loss (-)` is a gross profit, respectively a gross loss.

Table 4.4: ARIMA fitting results – Binance data

Time frame	ARIMA order	Number of orders	Number of valid orders
30 min	(2, 0, 2)	1	1
30 min	(0, 0, 3)	1	1
30 min	(0, 0, 1)	1	0
30 min	(0, 0, 2)	1	0
30 min	(0, 1, 5)	1	0
30 min	(1, 0, 1)	1	0
30 min	(2, 0, 3)	1	0
30 min	(2, 0, 1)	1	0
30 min	(1, 0, 4)	1	1
30 min	(4, 0, 4)	1	0
60 min	(4, 0, 5)	1	0
60 min	(5, 0, 0)	3	2
60 min	(0, 0, 2)	1	0
60 min	(0, 1, 4)	1	0
60 min	(2, 0, 3)	1	0
60 min	(0, 0, 1)	1	0
60 min	(1, 0, 2)	1	0
60 min	(0, 0, 0)	1	0
12 hours	(2, 0, 3)	2	0
12 hours	(2, 0, 0)	2	0
12 hours	(2, 0, 1)	2	0
12 hours	(2, 0, 2)	2	0
12 hours	(1, 0, 2)	1	0
12 hours	(0, 0, 2)	1	0
1 day	(5, 0, 0)	1	0

In the result tables, the order is one operation, buy or sell; a trade refers to one buy and sells operation together. Obviously, there is a relation $O = (T+) + (T-) + (T =)$, and rationally the number of orders should be always even. But in the tables, we can see that 0 is both, even and odd. This is because sometimes the last sell order is not executed that day and it remains open until the next day. In order to provide information about the profit, the last price of the day is used instead to count the profit/loss.

4.4.0.1 Notes

We recommend also seeing the notes in the subsection 3.5.

1. A commission of a trade and a spread (the difference between the ask and bid price) are not taken into account. The results of all strategies are gross. More

Table 4.5: Number of quotes/trades in specific time frames

Time frame	Tick (quotes number)	Binance (trades number)
10 s	20	102
1 min	121	615
5 min	605	3075
10 min	1211	6150
30 min	3635	18450
60 min	7270	36901

details are described in the subsection 4.4.1.

2. For backtesting with Tick data we use *middle price*, $(ask + bid)/2$, for simplification.

3. The amount of money with which the backtesting is carried out is USD 1,000 for Tick data, and USDT 1,000 for Binance data.

4. The results of strategies are written in the absolute form in currency USD for Tick data and in cryptocurrency USDT for Binance data.

4.4.1 Net/gross profit

When backtesting the strategies we do not take into account a commission for a broker or an exchange and a spread, thus the profit in the tables is gross. This is because each broker or exchange has different commissions and spreads. If we work with a specific commission or spread, then it is more difficult to get the net profit. The profit reduced by the commission can be calculated as:

$$\text{reduced profit} \approx \text{gross profit} - (\text{number of orders} \times \text{monetary amount} \times \text{commission}).$$

The sign denoting approximate equation is there for two reasons; the first is that sometimes it is not possible to execute an order in an exact amount of money and it is executed in the nearest possible amount, while the second is that a sell order usually realizes profit or loss, therefore the amount of money is greater or smaller than it was in a buy order. In both cases, the parameter that varies is the monetary amount.

Reducing the profit or loss by spread is not a trivial issue. Unlike the commission, which varies only among the exchanges, the spread can vary even during a day on one exchange. Thus, we will work only with the reduction by commission.

Table 4.6: Mean reversion strategy results – Tick data

Type of MA	O	T+	T-	T=	Profit (+) / Loss (-)
CMA	8	4	0	0	+6.95
SMA-121	1250	412	201	12	-2.56
SMA-605	439	171	47	1	+5.73
SMA-1211	207	76	26	1	+3.01
SMA-3635	80	30	9	1	+3.34
SMA-7270	46	20	3	0	+3.76
SMA-43620	8	3	1	0	+0.12
SMA-87240	2	1	0		+0.79

Table 4.7: Mean reversion strategy results – Binance data

Type of MA	O	T+	T-	T=	Profit (+) / Loss (-)
CMA	6	3	0	0	+9.92
SMA-615	1766	533	335	15	-47.81
SMA-3075	424	138	73	1	-30.58
SMA-6150	232	65	50	1	-14.36
SMA-18450	120	43	17	0	+4.44
SMA-36901	60	16	13	1	-3.57
SMA-221406	30	12	3	0	-4.49
SMA-442812	10	2	3	0	-6.65

4.4.2 Mean reversion strategy

The results for the mean reversion strategy are in Table 4.6 for Tick data and in Table 4.7 for Binance data. In column `Type of MA` is the type of used moving average. The type `CMA` is explained in the subsection 3.5.1.1. In the `SMA` cases, the number that is after `SMA-` is the size of the rolling window for the moving average in the unit of the number of quotes, resp. trades. The five last columns of the tables are already described at the beginning of section 4.4.

For both, Tick data and Binance data the best and worst results were achieved with the same moving averages. The mean reversion strategy is the most successful using the CMA and the least successful using SMA-121, resp. SMA-615 averages, which both refer to approximately a 1-minute time frame.

4.4.3 Momentum strategy

The results for the Momentum strategy are in Table 4.8 and in Table 4.9. The last five columns of the tables are already described at the beginning of section 4.4. Let us describe the first column `MA min-max-num`. The `min` parameter stands for the

Table 4.8: Momentum strategy results – Tick data

MA min-max-num	O	T+	T-	T=	Profit (+) / Loss (-)
10-100-5	1117	363	186	9	+4.94
10-100-10	745	242	126	4	+3.52
10-605-5	199	73	26	0	+9.10
10-605-10	115	37	20	0	+4.68
20-605-5	201	75	25	0	+9.14
20-605-10	119	40	19	0	+5.51
20-1211-5	91	34	11	0	+6.96
20-1211-10	60	22	8	0	+6.14
20-7270-5	19	6	3	0	+5.78
20-7270-10	9	3	1	0	+3.08
121-605-5	201	70	29	1	+6.86
121-605-10	149	52	22	0	+7.70
121-1211-5	101	39	11	0	+8.43
121-1211-10	64	23	9	0	+7.53
121-3635-5	35	14	3	0	+4.88
121-3635-10	15	6	1	0	+2.14
605-1211-5	131	43	22	0	+5.45
605-1211-10	114	41	16	0	+7.31
605-3635-5	33	11	5	0	+2.43
605-3635-10	17	7	1	0	+0.91

minimal length of a moving average, `max` stands for the maximal length of a moving average, and `num` is the number of moving averages in the closed interval `min-max`.

The best result for Tick data is USD 9.14 using five moving averages in the interval (20-605). For Binance data it is USDT 20.36 using ten moving averages in the interval (615-6150).

4.4.4 ARIMA strategy

The results for the ARIMA strategy are in Table 4.10 for Tick data and in Table 4.11 for Binance data. We tested the strategy with ARIMA(1, 0, 0) model using a 30-minute time frame (3635 values) for Tick data and with ARIMA(5, 0, 0) using a 60-minute time frame (36901 values) for Binance data. The orders of the models are the best results from ARIMA fitting; we can see the results in Table 4.3 and Table 4.4. The column `FC length` provides information about the length of the forecast from the model. The last five columns of the tables are already described at the beginning of section 4.4.

The best result is USD 1.61 for Tick data using a forecast of length 121 values (1-minute time frame). For Binance data it is USDT 2.84 using a forecast of length

Table 4.9: Momentum strategy results – Binance data

MA min-max-num	O	T+	T-	T=	Profit (+) / Loss (-)
102-615-5	890	272	172	1	-18.50
102-615-10	626	189	124	0	-27.30
102-3075-5	189	47	41	0	-14.14
102-3075-10	117	34	24	0	-18.48
615-3075-5	177	48	40	0	-16.80
615-3075-10	145	44	28	0	-0.98
615-6150-5	97	34	14	0	+12.65
615-6150-10	73	25	11	0	+20.36
615-18450-5	30	7	8	0	-3.07
615-18450-10	18	4	5	0	-9.34
3075-6150-5	127	41	22	0	+19.58
3075-6150-10	103	33	18	0	+14.65
3075-18450-5	34	10	7	0	+3.00
3075-1840-10	22	5	6	0	+5.85
6150-36901-5	12	2	4	0	-26.63
6150-36901-10	10	3	2	0	-16.77

Table 4.10: ARIMA strategy results – Tick data

FC lenght	O	T+	T-	T=	Profit (+) / Loss (-)
20	1698	388	417	44	+1.59
121	318	72	81	6	+1.61
605	80	20	19	1	+1.51
1211	48	12	12	0	0.30

Table 4.11: ARIMA strategy results – Binance data

FC lenght	O	T+	T-	T=	Profit (+) / Loss (-)
121	-	-	-	-	-
615	1111	234	204	117	+2.35
3075	227	40	50	23	+2.84
6150	117	18	25	15	-8.65

3075 values (5-minute time frame). The backtesting of the strategy using a forecast of the length of 121 values (20-second time frame) for Binance data lasted more than 24 hours (more than is the time interval of our data), therefore we interrupted the backesting process.

Table 4.12: Final comparison of best results of trading strategies

	Strategy	Order's number	Gross profit	Profit reduced by commission
Tick	Mean reversion	8	+6.95	+6.79
	Momentum	201	+9.14	+5.12
	ARIMA	318	+1.61	-4.75
Binance	Mean reversion	6	+9.92	+3.92
	Momentum	73	+20.36	-52.64
	ARIMA	227	+2.84	-224.16

4.4.5 Final comparison

In Table 4.12 there is a comparison of the best results of tested strategies based on the greatest gross profit. In order to have a better idea about the "net" profit (not taking the spread into account), we counted profit reduced by a commission. The reduced profit is in the column `Profit reduced by commission`. The size of the commission varies amongst exchanges and brokers.

For Tick data, we set the commission to 0.002 %. This is a commission offered by broker *Interactive Brokers*[4]. However, there is a condition that the minimal size of the commission is USD 2. If we have a smaller amount of money than USD 100,000, the commission will always be USD 2 and not 0.002 %.

For Binance data, we set the commission to 0.1 %. This is a standard commission offered by Binance[5]. There is no minimum size of the commission, it is always 0.1 %.

In the table, we can see how the number of orders can affect the final result. The best strategy according to gross profit is Momentum strategy using Binance data, whereas the best result according to the reduced profit is Mean reversion strategy using Tick data.

[4] Commissions Spot Currencies. *InteractiveBrokers* [online]. [Cit. 16.4.2023]. Available at: `https://www.interactivebrokers.com/en/pricing/commissions-spot-currencies.php?re=amer`

[5] Trading Fees. *Binance* [online]. [Cit. 16.4.2023]. Available at: `https://www.binance.com/en/fee/schedule`

Conclusion 5

In this thesis, we compared the strategy based on the forecast from the ARIMA model with two basic trading strategies based on the exploitation of moving averages. We worked with the high frequency trading data (tick data) of frequency of several values per second. Specifically, we used two datasets; the first contained trading quotes on the currency pair EUR/USD, and the second executed trades on the cryptocurrency pair BTC/USDT. Furthermore, we also researched the statistical distribution of the data.

We tried to find out if the IaT of data follow any statistical distribution. We found out that data does not follow any tested statistical distributions, not even on time frames of various lengths. Also, the data does not have the same statistical distribution during the day. For these reasons, we treated the data in terms of changes in price, not in terms of changes in time. If IaT had followed the exponential or power-law distributions, we could have used a continuous Hidden Markov model for making forecasts; as this was not the case, we could not use the Hidden Markov model.

We made a detailed description of the AR, MA, and ARIMA models, including the process of fitting the models, together with the needed properties of time series. The results of fitting the ARIMA model to data show that the ARIMA model is not a proper model for high frequency data. In the cases of both Tick and Binance data, the determined order heavily varied during the day and the majority of the determined orders were not a good fit. The best determined order of the ARIMA model for Tick data was (1,0,0) using a time frame of 30 minutes, and for Binance data it was (5,0,0) using a time frame of 60 minutes.

The trading strategies we chose for the purpose of comparison were the mean-reversion strategy and momentum strategy. The ARIMA model is a sophisticated statistical model; even though fitting the ARIMA model to data was not successful, we compared the ARIMA strategy with the chosen basic strategies. In Table 4.12, there is the final comparison between the best results of strategies. The monetary amount used for backtesting was USD 1,000 for Tick data and USDT 1,000 for Binance data. The results are written in the absolute form in USD for Tick data and

in USDT for Binance data. The considered commission for Tick data was 0.002 % per order conditioned by a trading amount greater than USD 100,000, and for Binance data it was 0.1 % without the condition. In the table, we can see that ARIMA strategy has worse results than Mean reversion and Momentum strategy for both Tick and Binance data. The strategy Mean reversion using Tick data has the best results.

Because the ARIMA model was not a good choice for the data, further research should consider implementing alternative techniques. Some alternatives may include: working with data of lower frequency, trying to work with realized volatility and not with logarithmic returns, choosing other statistical models, or considering the use of machine learning methods.

Bibliography

CARTEA, Álvaro; JAIMUNGAL, Sebastian; PENALVA, José, 2015. *Algorithmic and High-Frequency Trading*. Cambridge, UK: Cambridge University Press. ISBN 978-1-107-09114-6/hbk.

CIPRA, T., 2020. *Time Series in Economics and Finance*. 1st. Prague, CZ: Springer Cham. ISBN 978-3-030-46346-5.

DEGIANNAKIS, Stavros; FLOROS, Christos, 2015. *Modelling and Forecasting High Frequency Financial Data*. London, UK: Palgrave Macmillan. ISBN 978-1-137-39648-8/hbk; 978-1-349-56690-7/pbk; 978-1-137-39649-5/ebk. Available from DOI: 10.1057/9781137396495.

DODGE, Yadolah, 2008. *The concise encyclopedia of statistics: With 247 tables*. New York, NY, United States: Springer-Verlag New York Inc. ISBN 9780387317427.

GARNER, Carley, 2012. *Currency Trading in the FOREX and Futures Markets*. Upper Saddle River, NY: Pearson Education. ISBN 978-0-13-293137-3/hbk.

HALLS-MOORE, Michael L., 2016. *Advanced algorithmic trading*.

LUNGA, Jakub, 2020. *Využití Hidden Markov Modelů při predikci finančních časových řad*.

NAU, R., 2020. *Statistical forecasting: notes on regression and time series analysis*. Available also from: https://people.duke.edu/~rnau/411home.htm. Last accessed 6 March 2023.

OFFICE OF THE FEDERAL REGISTER, National Archives; RECORDS, 2010. *Federal Register Vol. 75, No.13, January 21, 2010* [Office of the Federal Register, National Archives and Records Administration]. Available also from: https://www.govinfo.gov/app/details/FR-2010-01-21.

REVENDA, Z., 2014. *Peněžní ekonomie a bankovnictví*. 5th. Prague, CZ: Management Press. ISBN 978-80-7261-279-6.

ROBERT H. SHUMWAY, David S. Stoffer, 2017. *Time Series Analysis and Its Applications*. 4th. Springer Cham. ISBN 978-3-319-52452-8.

SPYROS G. MAKRIDAKIS Steven C. Wheelwright, Rob J. Hyndman, 1998. *Forecasting: Methods and Applications*. 3rd. Wiley. ISBN 978-0-471-53233-0.

TABOGA, M., 2021. *"Statistical model", Lectures on probability theory and mathematical statistics* [Kindle Direct Publishing. Online appendix.]. Available also from: `https://www.statlect.com/glossary/statistical-model`. Last accessed 5 Januray 2023.

TSAY, Ruey S., 2005. *Analysis of Financial Time Series*. 2nd. Hoboken, New Jersey: John Wiley & Sons, Inc. ISBN 978-0-471-69074-0.

List of Figures

List of Tables

Appendix

In the following there is information about the folder structure of directory "Application_and_libraries" in the thesis's archive, and copied comments from the individual files.

Folders description

- `Data_preprocessing`. The codes in this directory prepare the data for the main scripts.
- `Distributions`. In the directory there are scripts related to statistical distributions.
- `Backtrader_implementation`. There are implemented strategies for backtesting.
- `Processing`. The directory contains main scripts that are not classified in other directories.
- `Notebooks`. The directory contains Jupyter notebooks, where there is for example detailed description of ARIMA fitting.
- `Plots`. This directory contains all the codes by which were generated the graphs in the thesis.
- `Ancilarry_code`. In the directory there are few short ancilarry codes.

List of files

Data preprocessing

`bin_add_header+indexing.py`

- It adds a header and indexing to the original file.

- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: None.
- Data file needed: `BTCUSDT-trades-2022-01-03_original.csv`.
- It creates files: `BTCUSDT-trades-2022-01-03.csv`.

bin_get_only_priceChanges.py

- This script creates the file, where there are only rows with price change. E.g. i …5000; i+1 …5000 –> in created file will be only case i.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: `add_header+indexing.py`.
- Data file needed: `BTCUSDT-trades-2022-01-03.csv`.
- It creates files: `BTCUSDT-trades-2022-01-03-onlyPriceChanges.csv`.

bin_interarrival_times.py

- Figure 3.4 on page 49, text is on page 48 in book Cartea A., Jaimungal S. and Penalva J. "Algorithmic and High-Frequency Trading (CRC, 2015).pdf"
- Calculates time difference between two trades + statistics stuff about that. Creates (data - percentil 0.95) histogram with time difference, and scatter plot with log time difference.
- In input dataset are only trades, where is change in price.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `add_header+indexing.py`,
  - `get_only_priceChanges.py`.
- Data file needed: `BTCUSDT-trades-2022-01-03-onlyPriceChanges.csv`.
- It creates files: `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.

bin_parseDatetime.py

- It converts datetime from unix timestamp to ISO 8601 in millisecond accuracy.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: `add_header+indexing.py`.
- Data file needed: `BTCUSDT-trades-2022-01-03.csv`.
- It creates files: `BTCUSDT-trades-2022-01-03_parsedDate.csv`.

tick_add_middle_price.py

- Add a middle price (Ask+Bid) / 2 to dataset.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `choose_datepart_from_original_data.py` or
  - `parseDate_orig_data.py`.
- Data file needed: e.g. `EURUSD_2019_01_03.csv`.
- It creates files: e.g. `EURUSD_2019_01_03.csv`.

tick_choose_datepart_from_original_data.py

- It chooses part from original data in the given time range. It sets columns, parses date, removes two empty columns.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: None.
- Data file needed: original data file, e.g. `EURUSD_GMT+0_NO-DST-2014-01012020.csv`.
- It creates files: e.g.`EURUSD_oneY-2019.csv`.

`tick_choose_most_frequented_days.py`

- Creates: csv with 10 most frequented days and csv with data of most frequented day.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `choose_datepart_from_original_data.py` or
  - `parseDate_orig_data.py`.
- Data file needed: e.g. `EURUSD_oneY-2019.csv`.
- It creates files: e.g. `EURUSD_2019_01_03.csv`.

`tick_get_only_priceChanges.py`

- This script creates three files:
  - there are only rows with price change in ask,
  - there are only rows with price change in bid,
  - there are only rows with price change in bid or ask.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `choose_datepart_from_original_data.py` or
  - `parseDate_orig_data.py`.
- Data file needed: e.g. `EURUSD_2019_01_03.csv`.
- It creates files: e.g. `EURUSD_2019_01_03_bothPriceChanges.csv`.

`tick_interarrival_times.py`

- Figure 3.4 on page 49, text is on page 48 in book Cartea A., Jaimungal S. and Penalva J. "Algorithmic and High-Frequency Trading (CRC, 2015).pdf".
- It calculates time difference between two trades + statistics stuff about that.
- In dataset there are only trades, where there is change in ask or in bid price.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:

  - `choose_datepart_from_original_data.py` or
  - `parseDate_orig_data.py`, and
  - `get_only_priceChanges.py`.

- Data file needed: e.g. `EURUSD_2019_01_03_bothPriceChanges.csv`.
- It creates files: e.g. `EURUSD_2019_01_03_interarrivalTimes.csv` or `EURUSD_2019_01_03_interarrival_times_diff.csv`.

`tick_parseDate_orig_data.py`

- It parses datetime of original data, drops two useless columns and result saves to csv file.
- Similar to the script `choose_datepart_from_original_data.py`, but it takes directly whole file.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: None.
- Data file needed: original data file, e.g. `EURUSD_GMT+0_NO-DST-2014-01012020.csv`.
- It creates files: e.g. `EURUSD_GMT+0_NO-DST-2014-01012020_parsedDatetime.csv`.

`replace_Nan.py`

- It replaces all NaN values with 0.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: None.
- Data file needed: any.
- It creates files: any.

Distributions

`bin_dist_expon.py`

- It tests if data of time frames follow exponential distribution. Lenght of time frames is given by parameter `t_delta`.
- It uses whole dataset and dataset cutted on treshold 935 ms. That is 95th percentile.

- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create Binance interarrival times.
- Data file needed: `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g. `BTCUSDT_expon_dist_5min.csv`.

`bin_dist_powerlaw.py`

- It tests if data of time frames follow powerlaw distribution. Lenght of time frames is given by parameter `t_delta`.
- It uses whole dataset and dataset cutted on treshold 935 ms. That is 95th percentile.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create inter-arrival times.
- Data file needed: `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g. `BTCUSDT_powerlaw_dist_5min.csv`.

`bin_two_sample_KS.py`

- It tests if data of time frames t+1 and t have the same distribution with use of KS two-sample test.
- It uses whole dataset and dataset cutted on treshold 935 ms. That is 95th percentile.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create Binance interarrival times.
- Data file needed: `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g. `BTCUSDT_same_dist_5min_2KS.csv`.

tick_dist_expon.py

- It tests if data of time frames follow exponential distribution. Lenght of time frames is given by parameter `t_delta`.
- It uses whole dataset and dataset cutted on treshold 2000 ms. That is 95th percentile.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create Tick interarrival times.
- Data file needed: `EURUSD_2019_01_03_interarrivalTimes.csv`.
- It creates files: `EURUSD_expon_dist_5min_1KS.csv`.

tick_dist_powerlaw.py

- It tests if data of time frames follow powerlaw distribution. Lenght of time frames is given by parameter `t_delta`.
- It uses whole dataset and dataset cutted on treshold 2000 ms. That is 95th percentile.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create Tick interarrival times.
- Data file needed: `EURUSD_2019_01_03_interarrivalTimes.csv`.
- It creates files: `EURUSD_pow_dist_5min_1KS.csv`.

tick_two_sample_KS.py

- It tests if data of time frames t+1 and t have the same distribution with use of KS two-sample test.
- It uses whole dataset and dataset cutted on treshold 2000 ms. That is 95th percentile.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create Tick interarrival times.

- Data file needed: `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g. `EURUSD_same_dist_5min_2KS.csv`.

distfit_distributions_80.py

- It is finding which distribution data fits. It uses 80 distributions.
- In dataset there are only trades, where there is change in price.
- It can use Freedman–Diaconis rule to determine number of bins.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create inter-arrival times for Tick and Binance data.
- Data file needed:
  - `EURUSD_2019_01_03_interarrivalTimes.csv` or
  - `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g.
  - `IAT_distfit_80_distributions_IQR.csv` or
  - `BTCUSDT_IAT_distfit_80_distributions_IQR.csv`.

fitter_distributions_80.py

- It is finding which distribution data fits. It uses 80 distributions.
- In dataset there are only trades, where there is change in price.
- It can use Freedman–Diaconis rule to determine number of bins.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create inter-arrival times for Tick and Binance data.
- Data file needed:
  - `EURUSD_2019_01_03_interarrivalTimes.csv` or
  - `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.
- It creates files: e.g.
  - `IAT_fitter_80_distributions_IQR.csv` or
  - `BTCUSDT_IAT_fitter_80_distributions_IQR.csv`.

Backtrader implementation

runnable_tick.py

- Main class for running strategies for Tick data.
- Parameters must be set in this file in part "PARAMETERS SETTING" and in indicators files.
- Scripts needed to run before: see `Data\_preprocessing` how to create needed files.
- Data file needed: `EURUSD_2019_01_03_without-ID.csv`
- It creates files: results and log file.

runnable_binance.py

- Main class for running strategies for Binance data.
- Parameters must be set in this file in part "PARAMETERS SETTING" and in indicators files.
- Scripts needed to run before: see `Data_preprocessing` how to create needed files.
- Data file needed: `BTCUSDT-trades-2022-01-03_parsedDate.csv`
- It creates files: results and log file.

strategy_momentum_MAR.py

- Momentum strategy (using moving average ribbons).
- Logic of strategy according to: `https://github.com/Auquan/Tutorials/blob/master/Momentum%20Strategies.ipynb`. Additional information: `https://www.investopedia.com/terms/m/movingaverageribbon.asp`.
- Code based on example strategy from Quickstart Guide on Backtrader site: `https://www.backtrader.com/docu/quickstart/quickstart/`
- Parameters to set: none.

strategy_mean_rev.py

- Mean reversion strategy.
- Logic is created according to: `https://github.com/Auquan/Tutorials/blob/master/Mean%20reversion.ipynb`
- Code based on the example from Quickstart Guide on Backtrader site: `https://www.backtrader.com/docu/quickstart/quickstart/`
- Parameters to set: none.

strategy_arima.py

- ARIMA strategy.
- A strategy that is based on forecast from ARIMA model. Very simply: if forecast <> 0 –> sell/buy.
- Forecast is created in `indicator_arima_fc.py` - more details there (there are even the parameters for forecast).
- Based on the example from Quickstart Guide on Backtrader site: `https://www.backtrader.com/docu/quickstart/quickstart/`
- Parameters to set: none.

indicator_zscore.py

- Z-Score indicator.
- Is calculates Z-score of actual price. We use the Z-score in mean reversion strategy. Std from numpy isn't used because of efectivity.
- It can be used with simple moving average or with cumulative moving average.
- Parameters must be set in this file in part "PARAMETERS SETTING".

indicator_MAR.py

- Moving average ribbon indicator.
- Momentum indicator based on shapes of ribbons. Ribbon is created by serie of SMAs with different lenghts.
- Detailed description of implementation on: `https://github.com/Auquan/Tutorials/blob/master/Measuring%20Momentum.ipynb`

- Additional information: `https://www.investopedia.com/terms/m/movingaverageribbon.asp`
- Parameters must be set in this file in part "PARAMETERS SETTING".

indicator_CMA.py

- Cumulative moving average indicator.
- It calculates cumulative moving average, i.e. calculates average from 0 to i. With every event, as time passes, i is incremented by 1 and thus the window size for average is bigger and bigger.
- Parameters setting: None.

indicator_arima_fc.py

- Indicator for ARIMA strategy
- It makes forecast of logarithm returns using ARIMA model.
- Order of ARIMA model must be determined before running this code, you must use the same data as here, even the same lenght. Tool for use: notebook `ARIMA_fitting.ipynb`.
- Parameters must be set in this file in part "PARAMETERS SETTING".

Processing

histogram_numPerYear.py

- Counts number of trades per year. Result is save to csv file in form of table: Year, Number of trades.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: None.
- Data file needed: original Tick data files, e.g. `EURUSD_GMT+0_NO-DST-2011-01012014.csv`.
- It creates files: e.g. `histogram_numPerYear_2011-2014.csv`.

`to_Yahoo_format.py`

- It creates Yahoo! formated data from tick data, i.e. Date, Open, High, Low, Close
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to get needed data file.
- Data file needed: `EURUSD_oneY_2019.csv`.
- It creates files: `EURUSD_oneY_Yformat.csv`.

`to_Yahoo_reduce_days.py`

- It reduces the number of days according to count of days in yahoo data using merge function. E.g. it kicks out weekends. In tick data is one day of weekend, in Yahoo data there is no weekend day.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: `to_yahoo_format.py`.
- Data file needed: `EURUSD_oneY_Yformat.csv` and `EURUSD_yahoo.csv`.
- It creates files: `EURUSD_oneY_Yformat_reduced_days.csv`.

`arima_exhausting_fitting.py`

- It fits ARIMA model to data of time frames of different lenght. Data of specific time frame are randomly chosen from a day 10 times. I.e. time frame 30 min 5000 trades, it randomly chooses data from 70000:75000, then fits, repeats 10 times.
- Time frames are given according to trades per time, here must be typed manually. Whole day is fitted in Notebooks.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `trades_per_time_mean.py` and see
  - `Data_preprocessing` how to get needed files.
- Data file needed:

  - `EURUSD_2019_01_03_without-ID.csv` or
  - `BTCUSDT-trades-2022-01-03_parsedDate.csv`.

- It creates files: e.g. `EURUSD_arima_exhFit_table.csv`.

`trades_per_time_mean.py`

- Calculates the mean of how many trades are in specific time frame (parameter `time_deltas`). E.g. result: in 10 minutes window is average number of trades 1500.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to get the needed data files.
- Data file needed:
  - `EURUSD_2019_01_03.csv` or
  - `BTCUSDT-trades-2022-01-03_parsedDate.csv`.
- It creates files:
  - `EURUSD_2019_01_03_trades_per_time` or
  - `BTCUSDT_trades_per_time.csv`.

Notebooks

`ARIMA_fitting.ipynb`

This file contains ARIMA fitting methods. It considers identifying the order of differencing, and model selection using package pmdarima, AIC, and Ljung-Box test. The practical foundation of this code is from the files: https://github.com/Auquan/Tutorials/blob/master/Time%20Series%20Analysis%20-%202.ipynb and https://github.com/Auquan/Tutorials/blob/master/Time%20Series%20Analysis%20-%203.ipynb.

`ARIMA_exhausting_fitting.ipynb`

This file determines the ARIMA order using the pmdarima package.

ARIMA_forecasting.ipynb

This file contains ARIMA forecasting. The practical foundation of this code is based on: https://github.com/Auquan/Tutorials/blob/master/Time%20Series%20Analysis%20-%202.ipynb and https://github.com/Auquan/Tutorials/blob/master/Time%20Series%20Analysis%20-%203.ipynb.

Plots

arima_acf+pacf_plot.py

- Script for the figures in the thesis: "ACF, PACF - logarithmic returns", "ACF, PACF - ARIMA(0,1,0)", "ACF, PACF - ARIMA(2,0,2)", and the last "ACF, PACF - ARIMA(3,0,2)".
- It plots ACF and PACF for the residuals of fitted ARIMA model.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create needed files.
- Data file needed: `EURUSD_2019_01_03_without-ID.csv`.
- It creates files: `acf.pdf`.

data_check_Yahoo-Tick_close.py

- Script for the figure "The difference between Tick and Yahoo! data in close price" in the thesis.
- It checks tick data with Yahoo! data in close price.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  1. `to_Yahoo_format.py`,
  2. `to_Yahoo_reduce_days.py`.
- Data file needed:
  - `EURUSD_yahoo.csv` and
  - `EURUSD_oneY_Yformat_reduced_days.csv`.
- It creates files: `check_Yahoo_closeP.pdf`.

histogram_numPerYear.py

- Script for the figure "Histogram with number of quotes per year" in the thesis.
- Plots histogram of number of trades per year.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: `histogram_numPerYear.py` over all of the original data files, then manually copy the results into one file.
- Data file needed: `histogram_numPerYear_2004-2020.csv` (the one manually created).
- It creates files: `histogram_numPerYear.pdf`.

ribbons.py

- Script for figure "Moving average ribbons" in the thesis.
- It generates picture of moving averages ribbons. The buy and sell signals are based on the fact obtained during backtesting the strategy.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before: see `Data_preprocessing` how to create needed data files.
- Data file needed: `EURUSD_2019_01_03_without-ID.csv`.
- It creates files: `ribbons.pdf`.

day_distr-fit_4-qqplots.py

- Script for the figure "Q-Q plots of the best-fitted distributions" in the thesis.
- Plots 4 qq-plots. Data: Tick-whole, Tick-treshold, Binance-whole, Binance-treshold data.
- For each data is plotted one distribution of the best fitted distribution from file: `distfit_distributions_80.py`.
- Parameters must be set in this file in part "PARAMETERS SETTING".
- Scripts needed to run before:
  - `distfit_distributions_80.py` and see

  - `Data_preprocessing` how to create interarrival times for Tick and Binance data.

- Data file needed: `EURUSD_2019_01_03_interarrivalTimes.csv` and `BTCUSDT-trades-2022-01-03-interarrivalTimes.csv`.

- It creates files: the result need to saved manually into png, pdf doesn't work.

`price_plot.py`

- Script for the Figure in the thesis.

- Plot price of data. Tick or Binance data.

- Parameters must be set in this file in part "PARAMETERS SETTING".

- Scripts needed to run before: see `Data_preprocessing` how to create needed data files.

- Data file needed:

  - `EURUSD_2019_01_03_without-ID.csv` or
  - `BTCUSDT-trades-2022-01-03_parsedDate.csv`.

- It creates files:

  - `EURUSD_price_2019_01_03.pdf` or
  - `BTCUSDT_price_2022_01_03.pdf`.

Ancilarry code

`get_data_info.py`

- Writes down info about data.

- Parameters must be set in this file in part "PARAMETERS SETTING".

- Data file needed: any file with header and indexing.

- It creates files: None.

drop_column.py

- It drops a column from a dataframe.
- Parameters must be set in this file in part “PARAMETERS SETTING”.
- Data file needed: any with a header, without indexing. Can be easily changed in the code.
- It creates files: result file.