Dynamic Network and Data Science Applications in Finance, Security and Genomics

by

Ranathungage Thimani Dananjana

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Statistics
University of Manitoba
Winnipeg

Abstract

Data Science studies are prolific in many application areas from health to finance and from supply chain to computer network security with varying objectives. This thesis is focusing on pattern mining and network analysis of data in some of these application areas. In the study of pattern mining, our focus is to compute waiting time in observing a desired pattern in three different application areas: patterns in DNA sequences, patterns in unauthorized access to computer systems and patterns of price rise and drops in stock prices. A novel fuzzy transition probability (TP) matrix is introduced, and a novel pattern mining algorithm is proposed for sequence data of any length. The proposed algorithm, which avoids the inversion of the pattern matrix, is applicable to Markov chains with huge state spaces. Furthermore, it is illustrated with real data that the incorporation of fuzzyness to the transition probability matrix is crucial in obtaining realistic forecasts. In the second study, we propose a novel method based on financial networks and their PageRank scores to form pairs to apply in pairs trading. In order to illustrate the practical performance of the proposed methods, algorithmic trading profits using the commonly used cointegration method are compared with the profits made by proposed correlation-based financial networks. The proposed method offers an advantage over the commonly used cointegration method by identifying more profitable trading pairs (stocks) for pairs trading.

Keywords: Markov chains, Fuzzy Stochastic Matrices, Pattern Mining Algorithm, Dynamic Trading Strategies, Financial Networks, PageRank Algorithm

Acknowledgment Page

Firstly, I would like to sincerely thank my supervisor, Dr. Aerambamoorthy Thavaneswaran and my co-supervisor, Dr Ruppa Thulasiram, for their guidance, unwavering support, and helpful advice throughout my Master's programme. Being supervised by them was an absolute honour.

I would also like to extend my sincere thanks to my advisory committee members, Dr. Alexandre Leblanc and Dr. Sumeet Kalia. I am deeply grateful for the time they dedicated to reviewing my work and for their insightful suggestions and comments, which significantly enhanced the quality of my research.

My sincere gratitude goes out to the staff, supporting staff, and my colleagues at the University of Manitoba, Department of Statistics. Furthermore, I am appreciative of the financial assistance that the University of Manitoba's Department of Statistics has kindly given over the previous two years.

Finally, I wish to convey my heartfelt appreciation to my family for their steadfast belief in me and their continual encouragement to follow my dreams. I am especially grateful to my dear husband, whose unwavering support and motivation have been crucial to my journey over the past two years.

This achievement would not have been possible without the assistance of each and every one of you.

Dedication Page

This work is dedicated to my mother, father, two brothers, and my beloved husband for their unconditional love and unwavering support provided to me throughout my academic journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In multidisciplinary domains, dynamic networks and sequence data pattern mining have proven to be indispensable tools. Finance is our main domain of concern when adopting these tools into practice. In addition, we considered genomics and computer security applications to further test the applicability of the proposed algorithms in pattern mining.

Analysing sequential data arising in different domains such as finance, genomics and computer security to mine for patterns of interest is one of our major goals. In this context, we have adopted the concepts of Markov chain modelling, which is frequently in use for studying patterns in sequence data, to introduce a novel fuzzy approach of pattern mining. Our main goal is to identify the time until a certain pattern first appears in a given series after accounting for the uncertainty associated with the probability estimates. We first focused on the finance domain and considered patterns of price movement (price rise or drop), as it serves as a potential strategy to make a buy/sell/hold

decision in tradings. As a result, we identified the number of days it takes to observe four consecutive price rises or drops by considering daily prices of stocks and cryptocurrencies extracted from Yahoo! Finance. The knowledge we gain here on the behaviour of financial markets is crucial when making timely decisions. We further apply the proposed algorithm in DNA sequence data and computer network data to identify time until patterns of choice emerge.

In the finance domain our next goal is to utilise a graph-based dynamic network approach to select asset pairs for trading. A financial network is built by connecting vertices or nodes (stocks) and edges (links). These networks are dynamic in the sense that stocks and the links connecting them (usually correlation or cointegration) can vary between time periods. The trading methods that we adopt are pairs trading and multiple trading of stocks. These trading mechanisms are referred to as algorithmic trading as they use computer algorithms to automate and execute tradings. The first step in these tradings is selecting assets to make buy or sell decisions. Cointegration theory has extensively been used in the literature for this purpose (Liang et al., 2020a; Liang et al., 2020b; Brunetti and De Luca, 2023). The same authors, (Brunetti and De Luca, 2023) discuss the superiority of using a distance based (minimising the sum of squared deviations between the normalised prices) method and Miao (2014) suggests ranking stock pairs according to the Pearson correlation coefficient to pre-select assets for trading. In our approach, a minimal spanning network based on threshold correlation is utilized to rank and pair stocks

for trading. The network graphs here act as the foundation to look into the hidden structures and behaviours of real-world financial stocks. We compare the trading profits made by pairs selected using the cointegration approach and those made by the proposed network approach.

Chapters 2 and 3 discuss how fuzzy pattern mining can be used to identify time until certain patterns emerge in sequence data and how dynamic networks and their ranking of nodes can be used to form profitable pairs in pairs trading respectively.

Chapter 2

Application of a Novel Fuzzy Pattern Mining Algorithm for Sequence Data

2.1 Introduction

Diverse applications of sequence data arise in various domains, ranging from biology and genetics to linguistics, finance, and information technology. For example, there is a growing interest in using Markov chain (MC) models in studying patterns in sequence data. Some well-known examples of sequence data are deoxyribonucleic acid (DNA) sequence data, trading signals generated by stock/cryptocurrency prices and alarm message sequences in security systems. In many practical applications, an MC model is first fitted to the sequence data, and then the transition probability (TP) matrix is calculated (Kimou et al., 2010 and Yang et al., 2020). Once the sequence data are presented, researchers

are more interested in certain specific patterns generated by Markov chains. The process of investigating and identifying underlying processes for specified patterns is known as pattern mining. The purpose of pattern mining in large data sets is to capture and compare trends that have previously been associated with observations such as diseases, trading, malicious events in finance, and computer network security.

One of the most widely used technologies in the biological sciences is gene sequencing. In genomics, with large genome sequencing projects becoming less expensive, pattern mining in such sequence data becomes an essential part of identifying potential diseases and treating an individual accordingly (Yang et al., 2020). Genomic data may include the sequence of nucleotides in DNA, as well as information about gene expression, regulation, and interactions. DNA is a long, double-stranded helicoidal structure found in the cells of living organisms and usually, a sequence fragment in a DNA sequence contains the genetic information that determines the characteristics and functions of an organism. Therefore, identifying patterns in DNA sequences plays a crucial role in DNA pattern mining.

Moreover, pattern mining being the process of examining a large batch of information to identify trends and patterns, it can be used for a variety of purposes, such as determining the buying preference of customers, detecting fraud and anomalies, and filtering spam. Pattern mining programs break down patterns and connections in data (Friedman, 1998) based on what information

users request or provide.

In order to determine whether a business is stable or profitable enough to make a capital investment, financial analysis of data is very important. Some of the main financial applications of data mining and machine learning include forecasting future financial events, such as stock markets and foreign exchange rates, predictive financial and investment analysis, trading futures, comprehending and managing financial risks, etc. As a high volume of financial data becomes available, data-driven modelling has become the most attractive technology for various financial applications (Thavaneswaran et al., 2022a). Pattern mining techniques can reveal hidden patterns and predict future trends/forecasts and behaviors in the financial markets. Mining data to generate trading signals and profits, especially with high-frequency data arising with cryptocurrencies, typically requires advanced statistical, mathematical, and artificial intelligence techniques. In algorithmic trading, trading signals are generated by identifying the patterns for stocks (Liang et al., 2020b and Thavaneswaran et al., 2020) and cryptocurrencies (Thavaneswaran et al., 2022b) to maximize the returns/profits.

Our objectives in this chapter include a) identifying a desired nucleotide pattern in a DNA sequence and where it appears in the sequence; b) implementing computer security systems (a honeypot) for information systems by identifying the riskiest network ports; c) avoiding potential risks of investments by monitoring for patterns in the log returns of stocks prices and maximizing

investment returns. The main contribution is to fit an appropriate MC model to a given data sequence and use the proposed pattern mining algorithm to obtain expected waiting times for MC-generated patterns.

2.2 Organization

Chapter 2 is organized as follows. Introduction to pattern mining and related work are summarized in Sections 2.1 and 2.3. Introduction to the fuzzy approach is discussed in Appendix A. Existing methods of pattern mining and their limitations are discussed in Section 2.4. Our proposed method of pattern mining and three data applications are described in Section 2.5. Finally, a conclusion for this Chapter is given in Section 2.6.

2.3 Related Work

Over the last few decades, we have seen remarkable advancements in biomedical research and biotechnology, accompanied by a significant surge in biomedical data. The challenge has shifted from simply amassing biomedical data to effectively extracting valuable insights and knowledge from it. In the literature, sequential pattern mining of DNA includes mining for repeated patterns using the Apriori algorithm as the traditional method (Yang et al., 2020). The Sequence Pattern Mining based on Markov chain (SPMM) algorithm, given in Junyan and Chenhui (2015), enhances the effectiveness of mining frequent patterns in DNA sequences. Markov chains are frequently used to model DNA

sequences. Avery (1987) used MC modelling to predict the occurrence of the sequence 'CTGAC' in the context of DNA.

Modern computing environments are mobile and dynamic. Therefore, to capture fraudulent activities in online banking and network applications, a dynamic pattern-mining procedure is required. Malicious attacks are constantly evolving (e.g., phishing attacks as in Sun et al., 2022), and the attack surface is also evolving with them. Therefore, the availability of data for predicting potential attack scenarios is essential for the risk assessment of computer systems in any network. Binyamini et al. (2021) presented an automated framework for modeling attack techniques from the text description of a security vulnerability. With such a framework, they could assess potential risks to the system. Several studies on countermeasures to such potential attacks were reported in the past (see for example, Xinming et al., 2005), and they are also subject to risk avoidance.

In finance, one traditional way to decide on buying or selling assets is through the mean reversion process, which is a theory that an asset's price will tend to converge to the average price over time. In other words, it is expected that variations from the average price would return to the average. Numerous trading strategies used in the financial market are built on this knowledge. There are multiple technical analyses in making buy or sell decisions (see Bakhach et al., 2018). Mining a pattern of price movement would serve as a potential strategy to make a buy/sell/hold decision, and the focus of our study

is on this application domain.

The proposed pattern mining algorithm is applied and tested on three applications in the domains of (1) finance to make trading decisions, (2) in genomics to identify waiting time for a desired pattern to first appear in a sequence and (3) in computer security to identify time to possible malicious attacks.

Appendix A provides an introduction to the fuzzy set theory.

2.4 Traditional Approach for Pattern Mining

2.4.1 Patterns in Sequences

A pattern is a sequence $(p_1, p_2, ..., p_n)$, such that each of the p_i is an element of the sample space S.

For example, consider flipping a coin with sample space $S = \{H, T\}$. Let the desired pattern to be observed be (H, H, H), getting three heads in a row. For $k = 1, \ldots, n$, let $s_k = (p_1, ..., p_k)$ be the subsequence consisting of the first k elements of the pattern. A Markov chain is constructed with state space $\{\phi, s_1, ..., s_n\}$, with ϕ as the initial state and s_n as the desired pattern or the absorbing state. Assuming a repeated sampling from S, the absorption time for the Markov chain is the same as the time until the pattern emerges. In the above coin flips example, successive trials are independent and identically distributed. However, this is not necessary.

For instances where there are independent trials, as in the coin flip example, there exist a closed form formula to calculate the expected waiting time to observe desired patterns for the first time. Let p be the probability of getting heads on a single toss, then the expected waiting time to get the first head (H) in a series of coin tosses is $1/p$. This can be explained by using the concept of geometric distribution. The geometric distribution models the number of failures (tails) before the first success (heads) in a series of independent Bernoulli trials (coin tosses). The expected value of a geometric distribution is $1/p$, hence, the expected waiting time to observe a head for the first time is $1/p$. The expected number of tosses needed to get HH for the first time is $1/p+1/p^2$ and expected number of tosses to get r successive heads ($HHH\ldots H$) for the first time is $1/p+1/p^2+1/p^3+\cdots+1/p^r$. However, for more complex patterns or those generated by Markov chains, there does not always exist a closed-form formula. Hence, for MC generated patterns, we often resort to numerical methods, simulations, or solving systems of equations to find the expected waiting time.

2.4.2 Expected Time to Absorption

Let P be a transition matrix. Then, when $P_{ll} = 1$, then state l is an absorbing state. A Markov chain, which has at least one absorbing state is called an absorbing Markov chain.

Consider an absorbing Markov chain on k states for which t states are

transient and $k-t$ states are absorbing. The states can be reordered, as in the canonical decomposition, with the transition matrix written in block matrix form, that is

$$P = \left(\begin{array}{c|c} Q & R \\ \hline 0 & I \end{array} \right),$$

where Q is a $t \times t$ matrix, R is a $t \times (k-t)$ matrix, 0 is a $(k-t) \times t$ matrix of 0's, and I is the $(k-t) \times (k-t)$ identity matrix. For an absorbing Markov chain with t transient states,

$$F = (I - Q)^{-1} \tag{2.1}$$

is known as the fundamental matrix, where F_{ij} is the expected number of steps to reach state j given that the chain starts in state i.

For an absorbing Markov chain starting in transient state i, let a_i be the expected absorption time or the expected number of steps to reach some absorbing state. The number of transitions from i to an absorbing state is simply the sum of the number of transitions from i to each of the transient states until eventual absorption. As the expected number of steps from i to transient state j is F_{ij}, it follows that

$$a_i = \sum_{k \in T} F_{ik},$$

where T is the set of transient states. In vector form, $a = \mathbf{F1}$, where $\mathbf{1}$ is the column vector of all 1s of length t. That is, the expected absorption times are the row sums of the fundamental matrix.

2.4.3 Algorithm for traditional approach

The first pattern mining approach we discuss in this study is the traditional approach. In many practical applications, an MC model is fitted first to the sequence data, and then the transition probability matrix is estimated. The traditional approach requires constructing a TP matrix and finding the inverse of the associated fundamental matrix (Equation 2.1). For convenience, we refer to the traditional approach as the matrix approach in this study. The matrix approach computes the expected waiting time of a desired pattern first appearing in a given sequence. The basic steps of the matrix approach are summarized in Algorithm 1.

The matrix approach requires a transition matrix with probabilities for the jumps from one state to another. The corresponding probabilities are estimated, and there is always uncertainty associated with these estimates. However, the matrix approach does not incorporate the uncertainty of probability estimates in the expected waiting time computation as no random error term is included in the model as explained in Dobrow (2016). Furthermore, the method requires computing the inverse of a matrix to find the expected waiting time. Thus, one can expect limitations in implementing the matrix approach in practice as there is always uncertainty associated with estimated probabilities, and finding

Algorithm 1 Matrix Approach

Require: Finite state space and corresponding transition probabilities from one state to another

1: Construct a transition probability (TP) matrix $\tilde{P}$ for the given state space.
2: Identify the desired pattern sequence of the states.
3: Consider a sequential search from the first state of the desired pattern until we reach the last state of the desired pattern.
4: Following the sequential search, construct a pattern matrix (TP matrix of the desired pattern) P with the desired pattern as an absorbing state.
5: Calculate the fundamental matrix (F) by inverting the matrix $(I - Q)$ where I is a identity matrix and Q is a minor matrix obtained from P after deleting last row and last column as there is only one absorbing state.
6: Calculate the Expected waiting time (expected number of data points until we reach the desired pattern for the first time) by adding first-row elements of F.

the inverse of the matrix may not always be possible. Hence, to overcome the limitations of the matrix approach, a novel data-driven fuzzy pattern mining algorithm is proposed in this study. We also extend our data-driven fuzzy pattern mining algorithm for both categorical sequences (e.g., DNA sequences) and numerical sequences (e.g., asset prices), and they are summarized in Algorithm 2 in Section 2.5.1 and Algorithm 3 in Section 2.5.2.

2.5 Novel Pattern Mining Algorithms

2.5.1 Application I: Successive Occurrences of DNA Nucleotides

Biological sequence data include three kinds of sequences: DNA sequences, RNA sequences and protein sequences. These essential biomolecules can be found

in all life forms on Earth. A DNA sequence refers to the specific arrangement of nucleotide bases in a molecule of DNA (deoxyribonucleic acid). DNA is a long, double-stranded helicoidal structure found in the cells of living organisms and contains the genetic information that determines the characteristics and functions of an organism. The building blocks of DNA are four different nucleotide bases: adenine (A), thymine (T), cytosine (C), and guanine (G). They represent four types of nucleotides or distinct base states based on which nitrogenous bases they contain (Gentleman and Mullin, 1989). Hence, there are four distinct base states in a DNA series. The sequence of these bases forms the genetic code, which provides instructions for the synthesis of proteins and the regulation of various biological processes. For a DNA sequence, its letter constitution is called the corresponding DNA pattern. For example, "ATGCTAG" is the corresponding DNA pattern of the sequence A, T, G, C, T, A, G. The order and combination of these bases in a DNA sequence carry the genetic information necessary for the formation and functioning of an organism. The length of a DNA pattern is the number of letters in it. For example, the length of the DNA pattern "ATGCTAG" is 7.

DNA sequencing techniques allow scientists to determine the exact order of nucleotide bases in a DNA molecule, enabling them to study and analyze genetic information, identify genes, study genetic variations, and gain insights into the molecular basis of various biological processes and diseases. The nucleotides are the building blocks of nucleic acids, and nucleic acids are most commonly known as RNA and DNA. Mainly there are two ways of absorbing nucleotides into

the body's cells: 1) diet and 2) synthesized in the liver from common nutrients. Nucleotides contain instructions for building an organism. Thus, extending the understanding of nucleotide sequences improves the knowledge of genetic functions. A certain sequence of nucleotides reveals how the other substances in the cells are controlled and for instance, the possibility of initiating cancers in some forms. For illustration purposes, here in this study, we discuss the expected waiting times or the average number of nucleotides needed, to reach the pattern ACCGC for the first time. We compare the expected waiting times for patterns computed using the matrix approach (Algorithm 1) and using our novel pattern mining approach with fuzzy matrices (Algorithm 2).

The steps of the proposed pattern mining algorithm for categorical sequences are given in Algorithm 2.

The R package `astsa` (Stoffer and Stoffer, 2023) contains several datasets covering different fields of study. This study investigates the dataset `bnrf1ebv` (Nucleotide sequence - BNRF1 Epstein-Barr), which contains the nucleotide sequence of the BNRF1 gene of the Epstein-Barr virus (EBV) with 3954 base pairs (bp). In `bnrf1ebv`, the format of the data is "AGAATTCGTCTT…", but represented as "131144234244…" where $1 = A, 2 = C, 3 = G$, and $4 = T$.

Following the steps in Algorithm 2 for categorical sequences and considering successive occurrences of DNA nucleotides for `bnrf1ebv`, a TP matrix ($\tilde{P}$) is

Algorithm 2 Data-Driven Fuzzy Pattern Mining Algorithm for Categorical Sequences

Require: Data: Categorical sequence data with finite state space. Parameters: Let k be the number of distinct states.

1: Identify the distinct element of the state space.
2: Use R function `markovchainFit` from the R package `Markovchain` Spedicato (2017) to estimate the transition probability (TP) matrix $\tilde{P}$.
3: Construct a stochastic matrix A where A is $k \times k$ matrix with $a_{ij} = 1/k$ for each $i, j = 1, \ldots, k$.
4: Calculate the alpha cuts of $\tilde{P}$. $P(\alpha) = \alpha * A + (1 - \alpha) * \tilde{P}$ where $\alpha \in [0, 1]$ $\{p_{ij}(\alpha) = \alpha * a_{ij} + (1 - \alpha) * \tilde{p}_{ij}\}$.
5: Simulate N distinct sequences with the given states until the desired pattern first appears. Record the number of steps $(n_l\,; l = 1, \ldots, N)$ until the desired pattern appears first for each of the sequences.
6: Calculate the expected waiting time by taking the average number of steps to reach the desired pattern $(\sum_{l=1}^{N} n_l/N)$.

obtained. The proposed pattern mining algorithm is a fuzzy approach as it incorporates the uncertainty associated with the probability estimates using α-cuts, i.e., instead of point estimates (when $\alpha = 0$), we are generating interval estimates. The initial $\tilde{P}$ matrix is not necessarily a stochastic matrix; hence, α-cuts are calculated in such a way that $P(\alpha)$ becomes a stochastic matrix. $\tilde{P}$ matrix is fuzzified as $P(\alpha) = \alpha * A + (1 - \alpha) * \tilde{P}$, by setting α as a value between 0 and 1, with increments of 0.001. Here, $P(\alpha)$ is the generated fuzzy stochastic matrix and the doubly stochastic matrix A, with elements $a_{ij} = 1/k$, ensures the stochasticity of the generated TP $(P(\alpha))$ matrix. For DNA data, k is 4 as there are four base states in a DNA series as A,C,G and T.

Figure 2.1 summarizes the base proportions of nucleotides of the BNRF1 gene of the Epstein-Barr virus. Note that in the Epstein-Barr virus gene

sequence, the base proportions of nucleotides C and G are high compared to the base proportions of nucleotides A and T.

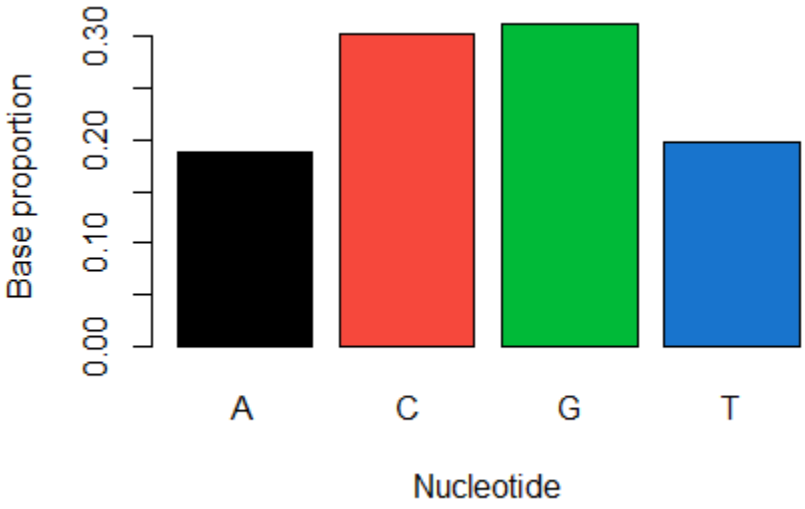


Figure 2.1: Base proportions of nucleotide in sequence data `bnrf1ebv`

A compound containing two nucleotides is known as a dinucleotide. Studying further, dinucleotides are important as they reveal more information about different functions of cells. or example, consider the functions of NAD+ (nicotinamide adenine dinucleotide). It is the most common and abundant molecule in single-cell organisms (e.g., bacteria) to multicellular organisms (e.g., primates). NAD+ helps convert food to energy and ensures cell functions are properly executed. If the cell functions are not properly executed, the bodies will be aging rapidly and exposed to diseases (Ying, 2006). For the nucleotide sequence of the BNRF1 gene of the Epstein-Barr virus, base proportions of dinucleotide are presented in Table 2.2 and visualized in Figure 2.2. It is

observed that the proportion for AA ($P(AA)$) is approximately 0.033, and it is very close to the proportion of $P(A)$'s squared. (Table 2.1: $P(A) * P(A) = 0.188 * 0.188$).

Table 2.1: Observed nucleotide proportions for `bnrf1ebv`

Nucleotide	Observed count	Observed proportion
A	744	0.188
C	1195	0.302
G	1232	0.312
T	783	0.198

Table 2.2: Observed dinucleotides proportions for `bnrf1ebv`

Dinucleotide	Observed count	Observed proportion
AA	130	0.033
AC	219	0.055
AG	247	0.062
AT	148	0.037
CA	283	0.072
CC	380	0.096
CG	262	0.066
CT	270	0.068
GA	244	0.062
GC	367	0.093
GG	417	0.105
GT	203	0.051
TA	86	0.022
TC	229	0.058
TG	306	0.077
TT	162	0.041

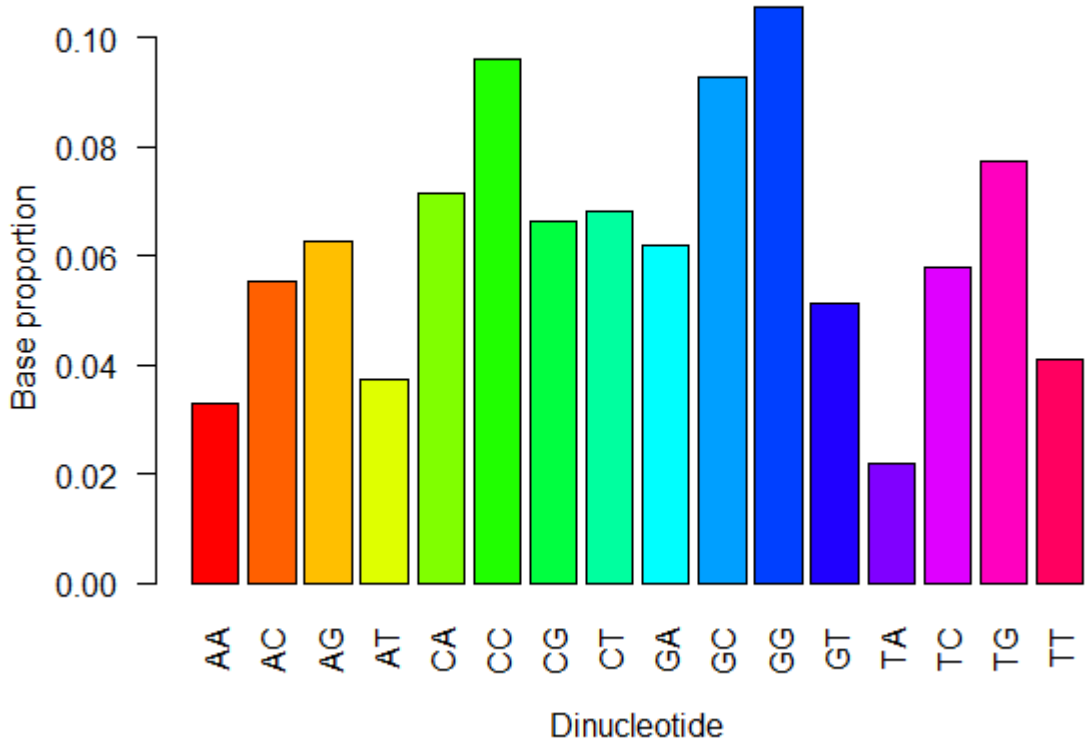


Figure 2.2: Base proportions of dinucleotide in sequence data `bnrf1ebv`

Using the R function `markovchainFit` and considering successive occurrences of DNA nucleotides, the following TP matrix ($\tilde{P}$) is obtained. Transition probabilities are estimated by dividing the number of transitions from state i to state j by the total number of transitions from state i. This yields a matrix where each entry p_{ij} represents the probability of transitioning from state i to state j.

$$\tilde{P} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0.175 & 0.294 & 0.332 & 0.199 \\ 0.237 & 0.318 & 0.219 & 0.226 \\ 0.198 & 0.298 & 0.339 & 0.165 \\ 0.110 & 0.292 & 0.391 & 0.207 \end{pmatrix} \end{array}$$

According to the matrix $\tilde{P}$, the probability of making a transition from nucleotide base A to nucleotide base C is 0.294 (1^{st} row 2^{nd} entry) in the `bnrf1ebv` sequence.

Following the transition matrix ($\tilde{P}$), an absorbing MC (P) is constructed for the patterns A, AC, ACC, ACCG, and absorbing state ACCGC. Note that starting state for the absorbing MC is the null set (ϕ). Figure 2.3 illustrates how the transition probabilities are assigned to each of the transition states in the absorbing MC.

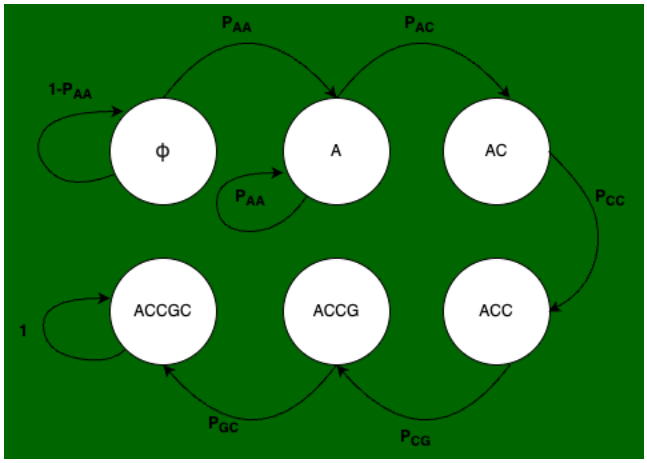


Figure 2.3: Transition probabilities for absorbing MC

The absorbing MC (P) matrix with absorbing state ACCGC is represented as,

$$P = \begin{array}{c} \\ \phi \\ A \\ AC \\ ACC \\ ACCG \\ ACCGC \end{array} \begin{array}{c} \begin{array}{cccccc} \phi & A & AC & ACC & ACCG & ACCGC \end{array} \\ \begin{pmatrix} 0.825 & 0.175 & 0 & 0 & 0 & 0 \\ 0.531 & 0.175 & 0.294 & 0 & 0 & 0 \\ 0.445 & 0.237 & 0 & 0.318 & 0 & 0 \\ 0.544 & 0.237 & 0 & 0 & 0.219 & 0 \\ 0.504 & 0.198 & 0 & 0 & 0 & 0.298 \\ 0 & 0 & 0 & 0 & 0 & 1.000 \end{pmatrix} \end{array}$$

For the matrix approach computations, we obtain matrix Q by deleting the last row and the last column of the P matrix (step 5 of Algorithm 1).

Observe that when the chain moves from state ACC to ACCG in matrix P, the corresponding TP is 0.219 (probability corresponds to the 4^{th} row and 5^{th} column of P). This probability of 0.219 is equal to the probability in the TP matrix ($\tilde{P}$) for moving from state C to G (2^{nd} row 3^{rd} column probability). Note that each jump corresponds to a one-step jump from one state to another. Thus, it can move from state ACC to state A (2^{nd} entry in the 4^{th} row), and it is not possible to move from state ACC to state AC (3^{rd} entry in the 4^{th} row), as it is a two-step jump. Hence, the corresponding probability is zero. The first column is completed with the complementary probability, making the absorbing matrix an MC.

The R function `markovchainFit` also produces confidence matrices of estimated transition probabilities. For the 95% confidence level, `lowerEndpointMatrix`

and `upperEndpointMatrix` are given by Matrices 2.2 and 2.3 respectively. It is of interest to note that for both lower and upper endpoint matrices, the row sums do not add to 1. Thus, both lower and upper endpoint matrices are not the TP matrices of an MC. Therefore, even though lower and upper matrices help to capture the uncertainty of the estimates by producing interval estimates, they are not useful to construct α-cuts of the TP estimates. In contrast, fuzzy matrices proposed in this study ($P(\alpha) = \alpha A + (1-\alpha)\tilde{P}$) always produce Markov chains, by incorporating the doubly stochastic matrix A, and can be used to construct α-cuts for a given transition matrix $\tilde{P}$.

Lower Endpoint Matrix of $\tilde{P}$:

$$\begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0.145 & 0.255 & 0.291 & 0.167 \\ 0.209 & 0.286 & 0.193 & 0.199 \\ 0.173 & 0.268 & 0.306 & 0.142 \\ 0.087 & 0.255 & 0.347 & 0.175 \end{pmatrix} \end{array} \tag{2.2}$$

Upper Endpoint Matrix of $\tilde{P}$:

$$\begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0.205 & 0.333 & 0.373 & 0.231 \\ 0.264 & 0.350 & 0.246 & 0.253 \\ 0.223 & 0.329 & 0.371 & 0.188 \\ 0.133 & 0.330 & 0.435 & 0.239 \end{pmatrix} \end{array} \tag{2.3}$$

According the matrix approach (Algorithm 1), the fundamental matrix F for the `bnrf1ebv` sequence is,

$$F = (I - Q)^{-1} = \begin{pmatrix} 682.84 & 163.89 & 48.19 & 15.32 & 3.35 \\ 677.13 & 163.89 & 48.19 & 15.32 & 3.36 \\ 666.81 & 160.49 & 48.19 & 15.32 & 3.36 \\ 636.68 & 153.20 & 45.04 & 15.32 & 3.36 \\ 478.22 & 115.05 & 33.83 & 10.76 & 3.36 \end{pmatrix}.$$

The sum of the first row of the fundamental matrix (F) is 913.60. Hence, the average number of nucleotides needed to reach the desired pattern ACCGC for the first time is 913.60. The proposed pattern mining algorithm with fuzzy matrices suggests a expected waiting time of 913.597 when α equals 0.289 and N equals 1000 (outputs of Algorithm 2). Note that the expected waiting times are close, and α is not equal to zero. Thus, incorporating stochastic variation in the TP estimates through fuzzy matrices, the new approach provides an alternative path to produce α-cuts for transition matrices. Table 2.3 summarizes observed counts computed from the sequence itself and estimated counts derived from the stationary probabilities of MC ($\tilde{P}$) after multiplying by the `bnrf1ebv` sequence length - 3954. Observe that the counts are very close, suggesting that for a long sequence, estimated counts reach observed counts.

2.5.2 Application II: Computer Security Applications

In computer security, a mechanism designed to identify, divert, or even counter-attack attempts of unauthorized access to information systems is known as a

Table 2.3: Observed and Estimated Counts of nucleotides for `bnrf1ebv`

	Nucleotide			
	A	C	G	T
Observed Counts	744	1195	1232	783
Stationary probabilities	0.188	0.302	0.312	0.198
Estimated Counts	743.214	1195.306	1232.316	783.164

honeypot (Spitzner, 2003). Intruders trying to gain access to protected systems are a new threat that needs innovative solutions in the information era. A study conducted by Kimou et al. (2010) observes data from the www.Leurre.com project that contains information about the number of observed attacks on different ports (e.g., 80 (HTTP - Hypertext Transfer Protocol), 135 (EPMAP - End Point Mapper), 139 (NETBIOS-SSN - Network Basic Input/Output System-Session Service), and 445 (MicroSoft-DS - Microsoft Directory Services)). The distinct states are represented here by the ports being attacked, along with the state that indicates no port is being attacked. Hence, there are five distinct states for the MC matrix $\tilde{P}$. These authors have considered the most attacked ports in their study and obtained the following Markov transition matrix ($\tilde{P}$) for weekly attacks:

$$\tilde{P} = \begin{array}{c} \\ 80 \\ 135 \\ 139 \\ 445 \\ \text{No Attack} \end{array} \begin{array}{c} \begin{array}{ccccc} 80 & 135 & 139 & 445 & \text{No Attack} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 8/13 & 3/13 & 1/13 & 1/13 \\ 1/16 & 3/16 & 3/8 & 1/4 & 1/8 \\ 0 & 1/11 & 4/11 & 5/11 & 1/11 \\ 0 & 1/8 & 1/2 & 1/8 & 1/4 \end{pmatrix} \end{array}.$$

According to the matrix $\tilde{P}$, the probability of making a transition from port 135 to port 80 is 0 (2^{nd} row 1^{st} entry). The probability of making a transition from port 80 to any other port is 0 as well, hence, the transition probability to "No Attack" is 1 (1^{st} row entries).

Once the transition probabilities are estimated, several thrilling questions arise. One may ask what is the expected waiting time for observing the event "No Attacks" for four consecutive weeks (for a month)? or what port is most likely to get attacked after four weeks when the initial state is "No Attack"? In other words, we are interested in unique patterns of transition matrix labels and their corresponding probabilities.

First, the matrix approach is used (Algorithm 1) to obtain the absorbing MC matrix (P) by considering "No Attacks (N)" for four consecutive weeks as the absorbing state (NNNN).

$$P = \begin{array}{c} \\ \phi \\ \text{N} \\ \text{NN} \\ \text{NNN} \\ \text{NNNN} \end{array} \begin{array}{c} \begin{array}{ccccc} \phi & \text{N} & \text{NN} & \text{NNN} & \text{NNNN} \end{array} \\ \begin{pmatrix} 3/4 & 1/4 & 0 & 0 & 0 \\ 3/4 & 0 & 1/4 & 0 & 0 \\ 3/4 & 0 & 0 & 1/4 & 0 \\ 3/4 & 0 & 0 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Observe that when the chain moves from state N to state NN, the corresponding TP is 1/4 (3^{rd} entry in the 2^{nd} row of P). This probability of 1/4 equals to the probability in the TP matrix ($\tilde{P}$) for moving from N to N (5^{th}

entry in the 5^{th} row). Each jump corresponds to a one-step jump from one state to another. Thus, it can move from state N to state NN and it is not possible to move from state N to state NNN (4^{th} entry in the 2^{nd} row), as it is a two-step jump. Hence, the corresponding probability is zero. The first column (ϕ) is completed with the complementary probability, making the absorbing matrix an MC. For the matrix approach computations, we obtained matrix Q by deleting the last row and the last column of the P. Then the computed fundamental matrix F is

$$F = (I - Q)^{-1} = \begin{pmatrix} 256 & 64 & 16 & 4 \\ 252 & 64 & 16 & 4 \\ 240 & 60 & 16 & 4 \\ 192 & 48 & 12 & 4 \end{pmatrix}.$$

The sum of the first row of the fundamental matrix (F) is 340. Hence, the expected waiting time is 340 weeks until we observe four consecutive "No Attacks" (i.e., no attacks for a month). On the other hand, the proposed novel pattern mining algorithm with fuzzy matrices (Algorithm 2), which avoids inverting larger order matrices, provides the expected waiting time for observing "No Attacks for a month" as 338.4 weeks with the optimal value of α as 0.391 (outputs of Algorithm 2). It is important to note that when expected waiting times are close, α is not equal to zero.

The MC below provides the four weeks ahead robust probabilistic forecasts

computed using matrix $P(\alpha)$, with α equals 0.391 as:

$$P^4(\alpha) = \begin{array}{c} \\ 80 \\ 135 \\ 139 \\ 445 \\ \text{No Attack} \end{array} \begin{array}{c} \begin{array}{ccccc} 80 & 135 & 139 & 445 & \text{No Attack} \end{array} \\ \begin{pmatrix} 0.089 & 0.217 & 0.282 & 0.204 & 0.208 \\ 0.089 & 0.226 & 0.279 & 0.201 & 0.206 \\ 0.089 & 0.219 & 0.282 & 0.204 & 0.207 \\ 0.089 & 0.217 & 0.282 & 0.205 & 0.207 \\ 0.089 & 0.218 & 0.282 & 0.204 & 0.207 \end{pmatrix} \end{array}.$$

Observe that in matrix $P^4(\alpha)$, starting from the state "No Attack", port 80 (HTTP) is least likely to get attacked (1^{st} entry of 5^{th} row), and port 139 (NETBIOS-SSN) is most likely to get attacked in 4 weeks with respective probabilities of 0.089 and 0.282 (1^{st} and 3^{rd} entries of 5^{th} row). For both cases α equals 0.391.

Note that each element in the stochastic matrix A (3^{rd} step of Algorithm 2) is the same $(1/k)$, and it corresponds to the TP when we do not hold any information about the likelihood of going from one state to another. Thus, with matrix A, we assume each transition has an equal chance. Matrix P (TP matrix with absorbing state) is fixed as we believe we have the perfect information (we know exact transition probabilities from $\tilde{P}$). In the proposed fuzzy approach, fuzzyfying the matrix $\tilde{P}$ allows us to account for the stochastic variation in the TP estimates and it provides fuzzy estimates of transition probabilities ($P(\alpha)$) and expected waiting times.

2.5.3 Application III: Financial Applications

Stock is a security that represents the ownership of a fraction of a corporation. Corporation offers stocks to raise funds for their corporation activities. Stocks of blue chip companies are typically traded at high prices in stock markets as stakeholders believe their values would further increase in the future. However, due to market fluctuation and different corporate actions such as high-frequency trading using algorithmic trading programs, stock values are expected to vary every second. Thus, observing and predicting the general direction of the price change is important for a good investment.

In our study, the daily adjusted closing prices of stocks/indexes (SP500-Standard and Poor's 500, VIX-CBOE Volatility Index, AAPL-Apple Inc., and GOOG-Alphabet Inc.) are considered. The adjusted closing price provides an accurate snapshot of the stock value after adjusting for various factors, such as dividend payouts. Furthermore, we extend our numerical study by considering six cryptocurrencies (BTC-Bitcoin, ETH-Ethereum, BNB-Binance Coin, XRP-Ripple, DOGE-Dogecoin, and ADA-Cardano) based on their popularity and market capital. Cryptocurrencies are digital currencies that operate in a decentralized manner, with their values primarily determined by the trust users place in them (Bowala et al., 2022). Stock and cryptocurrency prices continuously change over time. Profitable stocks/cryptocurrencies in one period may lead to a loss in investment in another period due to price changes. Thus, efficient pattern prediction techniques could lead to profitable investments by

buying/selling stocks/cryptocurrencies at the right time.

Algorithm 3 Data-Driven Fuzzy Pattern Mining Algorithm for Numerical Sequences

Require: Data: Adjusted closing price of stocks/indexes and Cryptocurrencies $(S_t, t = 0, \ldots, n)$. Parameters: Let k be the number of distinct states.

1: $r_t \leftarrow \log S_t - \log S_{t-1}\,;\, t = 1, \ldots, n$
2: signals $\leftarrow$ 1 if $r_t \geq 0$ o.w. signals $\leftarrow$ -1 {A dataframe called signals is created according to the symbol of r_t values}.
3: Use R function `markovchainFit` from the R package `Markovchain` Spedicato (2017) to estimate the transition probability (TP) matrix $\tilde{P}$.
4: Construct a stochastic matrix A, where A is $k \times k$ matrix with $a_{ij} = 1/k$, here $k = 2$ for each $i, j = 1, 2$.
5: Calculate the alpha cuts of $\tilde{P}$. $P(\alpha) = \alpha * A + (1 - \alpha) * \tilde{P}$ where $\alpha \in [0, 1]$ $\{p_{ij}(\alpha) = \alpha * a_{ij} + (1 - \alpha) * \tilde{p}_{ij}\}$.
6: Simulate distinct N number of sequences with given states until the desired pattern first appears. Record the number of steps $(n_l\,;\, l = 1, \ldots, N)$ until the desired pattern appears first for each of the sequences.
7: Calculate the expected waiting time by taking the average number of steps to reach the desired pattern $(\sum_{l=1}^{N} n_l/N)$.

In this study, we estimate the expected waiting time until patterns of length four (four days) are reached using Algorithm 3. We generate a binary signal by observing the sign of the log return of adjusted closing prices (see Algorithm 3). If the log return is positive, the generated signal is 1, and if the log return is negative, the generated signal is 0. Hence, k here equals to 2. A pattern of "1111" corresponds to four consecutive days of price increase and a pattern of "0000" corresponds to four consecutive days of price drops. Our focus on this study is to compute the expected time to see each pattern of interest. We have considered daily adjusted closing prices from 2021-11-26 to

2022-11-26 for cryptocurrencies and from 2017-01-01 to 2022-11-30 for stocks. For cryptocurrencies, adjusted closing prices are available throughout the week. However, adjusted closing prices are only available on weekdays for stocks as the stock markets only operate during the weekdays. Hence, one month covers around 20 days of daily data for stocks.

A transition probability matrix ($\tilde{P}$) of the two states (state 1 ($r_t \geq 0$) and state -1 ($r_t < 0$) can be defined by reporting only the diagonal elements (p_{00}, p_{11}). Off-diagonal elements (p_{01}, p_{10}) can be calculated by considering the complement of diagonal transition probabilities ($p_{01} = 1 - p_{00}$, $p_{10} = 1 - p_{11}$). Hence, we can report complete transition probabilities for all the states (signals) of stocks/indexes and cryptocurrencies by reporting only the diagonal elements. Diagonal elements for the selected stocks/indexes and cryptocurrencies are: SP500 (0.431, 0.538), VIX (0.547, 0.426), GOOG (0.463, 0.544), AAPL (0.448, 0.521), BTC (0.542, 0.483), ETH (0.526, 0.471), BNB (0.473, 0.437), XRP (0.454, 0.439), DOGE (0.455, 0.420) and ADA (0.474, 0.409).

Table 2.4 and 2.5 summarize the expected waiting time (in days) using the novel algorithm for each of the patterns of length four for stocks/indexes (SP500, VIX, and GOOG) and cryptocurrencies (BTC, ETH, and BNB). Optimal α values that give an expected waiting time closer to the matrix approach for each pattern are also reported in the two tables for $N = 1000$. It can be seen that α changes with the patterns. This illustrates the importance of the new fuzzy approach to model risk of probability estimates. Among the stocks/indexes,

SP500 and Google have the highest expected waiting time to reach the pattern "0000"(49 and 41 days respectively). VIX has the highest expected waiting time for the pattern "1111" (47 days) and only 24 days until the pattern "0000". Thus, a four-day price decline ("0000") for VIX is expected to occur sooner than a four-day price increase ("1111"). Further, a price increase ("1111") for four consecutive days is observable in around one month (24 and 23 days) for both SP500 and Google.

Table 2.4: Fuzzy expected waiting time with α for SP500, VIX, and GOOG

	SP500			VIX			GOOG		
Pattern	Waiting Time*	Waiting Time**	α	Waiting Time*	Waiting Time**	α	Waiting Time*	Waiting Time**	α
0000	48.79	46.84	0.16	23.81	23.65	0.23	40.38	38.46	0.13
0001	21.95	21.55	0.15	13.49	13.50	0.21	18.76	18.77	0.12
0010	22.64	22.15	0.11	14.60	14.66	0.26	21.24	21.26	0.21
0100	22.44	22.20	0.19	14.52	14.44	0.19	21.28	21.22	0.27
1000	20.80	20.67	0.16	13.61	13.60	0.10	19.36	18.93	0.15
0011	17.45	17.26	0.14	17.20	17.19	0.17	16.00	16.00	0.26
0101	19.32	19.37	0.27	19.13	19.13	0.29	20.40	20.35	0.27
1001	17.59	17.62	0.25	17.94	17.92	0.13	18.14	18.14	0.13
0110	18.12	18.12	0.16	17.92	17.92	0.13	18.49	18.41	0.26
1010	18.40	18.47	0.10	18.87	18.88	0.17	20.45	20.45	0.10
1100	16.91	16.92	0.30	16.95	16.96	0.15	16.10	16.10	0.13
0111	14.07	14.14	0.26	21.03	21.05	0.23	13.55	13.49	0.16
1110	13.55	13.55	0.11	21.76	21.55	0.11	13.70	13.75	0.19
1101	14.57	14.54	0.12	22.67	23.11	0.13	15.74	15.74	0.23
1011	17.94	16.15	0.25	22.67	22.31	0.18	15.74	15.68	0.24
1111	24.11	24.09	0.22	46.57	46.49	0.23	22.64	23.20	0.10

Waiting Time* - Expected waiting time using matrix approach
Waiting Time** - Expected waiting time using fuzzy approach

Table 2.5: Fuzzy expected waiting time with α for BTC, ETH and BNB

	BTC			ETH			BNB		
Pattern	Waiting Time*	Waiting Time**	α	Waiting Time*	Waiting Time**	α	Waiting Time*	Waiting Time**	α
0000	23.11	24.66	0.20	25.45	25.54	0.21	36.01	34.72	0.1
0001	13.71	14.18	0.10	14.50	14.90	0.54	17.93	16.94	0.10
0010	16.31	16.75	0.87	16.30	16.41	0.10	16.84	16.71	0.32
0100	19.88	18.26	0.50	19.81	18.13	0.87	16.46	16.34	0.10
1000	14.20	13.97	0.10	14.42	14.68	0.32	16.34	16.36	0.14
0011	16.41	16.25	0.54	16.78	16.70	0.65	18.20	18.70	0.11
0101	21.10	21.02	0.20	19.78	19.78	0.76	17.23	17.22	0.21
1001	18.55	18.35	0.10	18.11	18.08	0.10	16.60	16.66	0.43
0110	18.15	18.68	0.50	17.94	17.95	0.43	17.30	17.15	0.65
1010	20.98	20.84	0.32	19.81	19.71	0.43	15.94	16.36	0.10
1100	17.57	16.62	0.54	16.15	16.11	0.21	19.88	18.23	0.21
0111	17.64	17.98	0.10	18.24	18.05	0.10	19.80	19.98	0.10
1110	17.65	17.46	0.21	18.03	18.42	0.10	19.23	19.16	0.10
1101	20.80	20.83	0.10	20.03	20.32	0.14	17.85	17.88	0.21
1011	20.80	20.21	0.11	20.03	19.89	0.21	17.85	17.84	0.21
1111	34.95	33.44	0.43	37.04	35.67	0.21	43.06	42.03	0.10

Waiting Time* - Expected waiting time using matrix approach
Waiting Time** - Expected waiting time using fuzzy approach

Similar to VIX, for BTC, ETH, and BNB (Table 2.5), the highest expected time is for the pattern "1111". Therefore, a consecutive price increase for four days takes longer time for BTC, ETH, and BNB than a consecutive price drop for four days.

In order to illustrate further, Table 2.6 shows the expected waiting time (in days) to observe the pattern "0101" for stocks/indexes and cryptocurrencies under both approaches (Algorithm 1 and Algorithm 3). We considered the pattern "0101" as it is important to identify potential reversals in market trends to make timely decisions. Observe that α cut values are non-zero at this point, hence, the importance of fuzzifying the transition probability matrix is

Table 2.6: Expected waiting time for the pattern 0101 using matrix and fuzzy approaches

		α	Waiting Time*	Waiting Time**
Stock/Index	SP500	0.27	19.32	19.37
	VIX	0.29	19.13	19.13
	GOOG	0.27	20.40	20.35
	AAPL	0.30	19.07	19.07
Cryptocurrency	BTC	0.20	21.10	21.02
	ETH	0.76	19.78	19.78
	BNB	0.21	17.23	17.22
	XRP	0.20	16.44	16.15
	DOGE	0.30	15.92	15.99
	ADA	0.20	16.12	16.05

Waiting Time* - Expected waiting time using matrix approach
Waiting Time** - Expected waiting time using fuzzy approach

established.

2.6 Conclusions

In many applications such as network security, healthcare, and finance, the state spaces of Markov chains can be very large, rendering conventional matrix methods impractical or even infeasible to deploy. Our work presents a novel fuzzy TP matrix and a novel pattern mining algorithm. The proposed algorithm, which avoids the inversion of the pattern matrix, is applicable to Markov chains in a wider context with huge state spaces. Unlike the existing work, the driving idea is that the resilient pattern mining algorithm is obtained by fuzzifying the TP matrix. The main contribution of this work is to fit

an appropriate MC model to a given data sequence and use the proposed fuzzy pattern mining algorithm to obtain robust probabilistic forecasts and expected waiting time for patterns. Three applications (DNA sequence data, MC model for network security, and patterns generated by the log-returns of the stocks/cryptocurrencies) of the proposed algorithm are discussed in detail.

Chapter 3

Novel Data-Driven Dynamic Network Science Application in Algorithmic Trading

3.1 Introduction

One of the recent developments in computational finance has been the rise in using graph-based approaches to analyse stock market dynamics systematically. In the algorithmic trading literature, stocks are commonly selected for pairs trading and multiple trading by using Engle-Granger and Johansen cointegration tests. In large datasets, identifying all pairs eligible for trading entails a significant computational burden. In this study, price correlation-based dynamic networks and differenced price series based correlation networks, as well as their nodes' importance ranks, are used to select pairs for trading. We compare the profitability of algorithmic trading strategies based on traditional

cointegration method with those derived from our proposed correlation-based financial networks.

Unlike the existing work, the novelty of the work is using a data-driven correlation-based financial network approach to propose stocks selection method for pairs trading. In this work, profit per transaction is used as a metric to compare the trading strategies. The superiority of the financial network stock selection method is discussed in detail.

3.2 Organization

Chapter 3 is organized as follows. An overview of algorithmic trading and its use for pairs and multiple trading is discussed in Section 3.4. The cointegration and network approaches of stock selection are explained in Section 3.5. Financial network theory and use of PageRank for pairs formation is discussed under the Sections 3.5.2 and 3.5.3 respectively. Next, Kalman filtering theory and algorithms used in pairs trading are discussed in Sections 3.5.4 to 3.5.8. Finally, data applications and conclusions are presented in Sections 3.6 and 3.7, respectively.

3.3 Related work

In the world of financial markets, pairs trading has become a well-known strategy that has drawn interest from both practitioners and scholars. Finding stock pairs that show cointegration—a sign of a possible long-term relationship

between their prices—is an essential first step in putting the pairs trading strategy into practice. Several studies including, Liang et al. (2020a), Liang et al. (2020), Liang et al. (2022) and Brunetti and De Luca (2023) use Engle-Granger (Engle and Granger, 1987) and Johansen (Johansen, 1991) tests of cointegration to pre-select stocks for pairs trading. Identifying all pairs eligible for trading in large datasets, as confirmed by Huck and Afawubo (2015), entails a significant computational burden. Therefore, Brunetti and De Luca (2023) use seven different methods including cointegration, distance and association-based methods to pre-select the assets for pairs trading. Gatev et al. (2006) were among the first to apply pairs trading using the distance approach. The strategy is implemented in two phases, as the authors describe: (1) during the formation phase, pairs are chosen by minimising the sum of squared deviations between their normalised prices; and (2) during the trading phase, a position is opened if the difference between the normalised prices diverges beyond a predetermined threshold. Gatev et al. (2006) evaluate the profitability of the distance-based pairs trading strategy by analysing the liquid US stocks in the Center for Research in Security Prices (CRSP) during the period from 1962 to 2002. They demonstrated that the strategy generates significant excess returns that continue to exist even after accounting for trading expenses. Numerous other empirical contributions, such as Do and Faff (2010) and Huck (2013), use the exact same methodology. While the latter examines the sensitivity of the profitability found by Gatev et al. (2006) under various parameterizations in

terms of the length of the formation period and of the opening threshold, the former demonstrates that the profitability found by Gatev et al. (2006) decreases if the analysis is extended to 2009. Moreover, the distance-based method may not ensure the mean reversion of price differences, nor can it capture the long-run equilibrium relationship between prices, which are mitigated by the cointegration approach.

Chen et al. (2019) choose the 50 most correlated stocks for each asset to create an equally weighted portfolio using the Pearson correlation coefficient (Freedman et al., 2007) between returns in univariate and quasi-multivariate settings. Pre-selecting assets prior to cointegration testing is one way that some recent contributions have tried to lessen the computational load associated with cointegration tests. Miao (2014) suggests ranking stock pairs according to the Pearson correlation coefficient of the prices and only testing for cointegration of those with a correlation of at least 0.9. This logically reduces the actual number of cointegration tests needed to implement the pairs trading.

Motivated by these manuscripts, we propose price correlation-based dynamic networks and differenced price series-based correlation networks and their importance ranks to select pairs for trading. In our approach, we first create financial networks based on the Pearson's correlations between price series (P_t) and on the Exponentially Weighted Moving Average (EWMA) correlation between first differenced (d_t) series. Then a minimal spanning network is obtained by setting a threshold for the correlation coefficients. These minimal

networks are then used in our approach to rank the nodes (stocks) and form pairs to apply and generate profits in pairs trading.

3.4 Algorithmic trading

Trading strategies are specific approaches that traders employ in the financial markets when deciding whether to buy (long) or sell (short) assets. This chapter is focused on a popular method in trading strategies in today's world, which is algorithmic trading for pairs trading. Algorithmic trading, often referred to as algo-trading (Cartea et al., 2015), is a trading strategy that uses computer algorithms to automate and execute a sequence of trading orders in financial markets. Algo-trading relies on pre-defined rules and criteria to make trading decisions, removing much of the human subjectivity from the process. Algorithmic trading aims to reduce trading costs, increase transaction execution speed and accuracy, and capitalise on market flaws. It can be used in various financial markets, including stocks, bonds, commodities, forex, and cryptocurrencies.

3.4.1 Pairs Trading

Pairs trading is a well-known trading strategy that was initially introduced in the mid-'80s by a quantitative group on Wall Street. Their primary objective was to exploit market inefficiencies by identifying statistical arbitrage opportunities in the equity markets.

Algorithmic trading for pairs trading is a quantitative trading strategy that involves the simultaneous purchase (long) of one asset and the sale (short) of another asset. The money received from shorting one asset is used to buy the other, requiring little or no initial capital. Hence, it is a zero-cost (self-financing) long-short portfolio of two assets. This trading strategy requires ongoing monitoring and adjustment, because it relies on the forecast of long-term mean reversion of the price spread between the two assets. Once the spread between the prices of the two assets goes above a certain upper level, the more expensive asset is sold, and the cheaper one is purchased. Later, if the spread declines below a certain lower level, the trade is closed by closing the short sale of the originally more expensive asset and selling the cheaper one. The lower and upper threshold levels can be either ad hoc (such as 1-sigma or 2-sigma thresholds) or data-driven as in our study. This is the basic process of pairs trading, as illustrated hypothetically by Figure 3.1. However, in real world as illustrated in Figures 3.4, 3.5 and 3.6, pairs trading is a sophisticated investment strategy that carries its own risks and may not always perform as expected.

3.5 Methodology

Implementation of algorithmic trading for pairs trading includes several steps. First, a pair of assets that moves together in the long run is identified. For instance, we may combine two equities from the same sector of the economy,

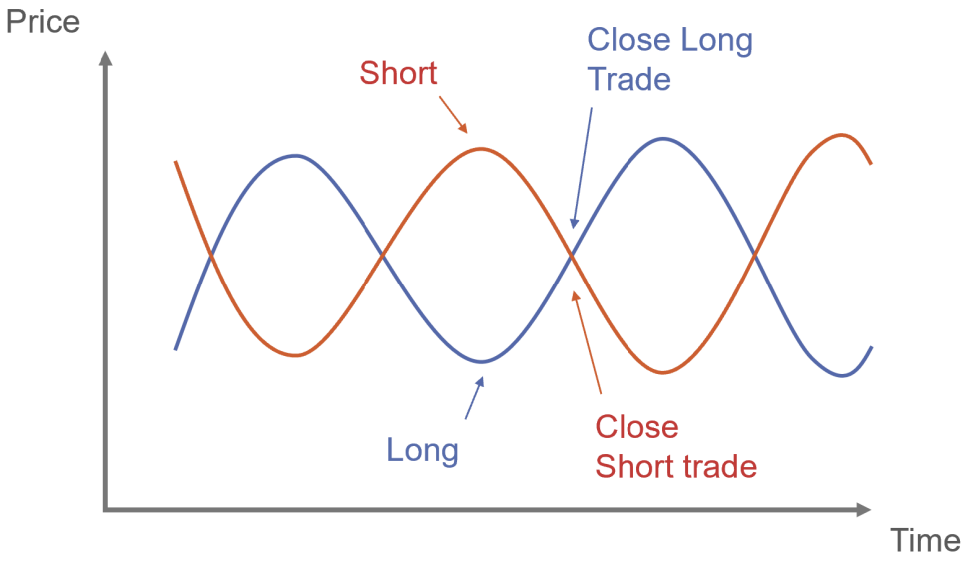


Figure 3.1: Idea of pairs trading

Source: https://algotrading101.com/learn/pairs-trading-guide/

two exchange-traded funds tracking the same commodity, or two closely related currency pairs. In this study, the concept of financial networks and the cointegration approach were used for the formation of pairs. In order to detect the cointegrated behaviour and network between the assets, historical price data were extracted from Yahoo! Finance. The final step is the use of trading algorithms to identify best-traded pairs in terms of their profits. However, the pairs that generate higher cumulative profits change over short periods and are

affected by transaction costs. In our approach, we use 18 overlapping windows of size 180 days spanning from February 2017 to December 2023 to identify trading strategies which can provide higher per-transaction profits.

3.5.1 Cointegration approach for pairs selection

Most of the economic and financial time series have a non-stationary behavior (Yang et al., 2014). Cointegration is a statistical concept that describes a long-term equilibrium relationship between two or more time series, such as pairs of stocks, bonds, foreign exchanges, or commodities. A group of non-stationary time series is said to be cointegrated if a linear combination of these time series results in a stationary time series. Cointegrated time series variables display a long-term equilibrium or common trend, despite short-term fluctuations or divergences. As assets that are cointegrated tend to converge to their equilibrium level over time, the concept of cointegration serves as the foundation for pairs trading strategies. When there is a brief divergence in the prices of the two stocks, pairs traders may enter positions in anticipation of a price reversion to equilibrium. In our study, we used the Engle-Granger test (Engle and Granger, 1987) and Johansen test (Johansen, 1991) to detect cointegration between a pair of assets and cointegration between multiple assets respectively (see Flori and Regoli, 2021; Liang et al., 2020a).

Engle-Granger (EG) Test

Cointegration and its application in pairs trading were made possible by the groundbreaking work of Engle and Granger (1987). Their study demonstrated how profitable it can be to take advantage of the mean-reverting behaviour of stock pairs that show cointegration. The Engle-Granger cointegration test considers the case that there is a single cointegrating vector. The test is based on a very simple idea that the residual of the cointegrating regression should be stationary if the variables are cointegrated.

The Engle-Granger cointegration test is a two-step procedure, wherein the first step, a regression is run to determine the residuals. Then, in the second step, stationarity of residuals is tested. If they are stationary, it suggests cointegration. A time series is considered non-stationary when unit roots are present. In the second step, the test searches the residuals of the regression for unit roots. This is often based on the Augmented Dickey-Fuller (ADF) or Dickey-Fuller (DF) (Dickey and Fuller, 1979) tests. However, the Engle-Granger test is only suitable for testing cointegration between two variables. For cointegration analysis involving multiple variables, the Johansen cointegration test is commonly used.

Johansen Test

The Johansen procedure, also known as the Johansen test, is a more thorough methodology that was introduced by (Johansen, 1991) and builds upon the

groundbreaking work of (Engle and Granger, 1987). The Johansen test is used for analyzing cointegration among multiple time series. This process is useful for choosing cointegrated pairs in pairs trading strategies because it provides a strong framework for calculating the number of cointegrating relationships among a set of variables. Vector autoregressive (VAR) models (Sims, 1980), which capture the joint dynamics of multiple variables, are the foundation of the Johansen test. This test determines the presence of cointegration and estimates the number of cointegrating vectors, or the rank of cointegration, through likelihood ratio tests. There are two different approaches to Johansen's test: the maximum eigenvalue method and the trace method based on linear algebra. Both forms of the test will determine if cointegration is present. The null hypothesis for both forms of test is that there are no cointegrating equations. The difference lies in the alternate hypothesis, where the trace test, which is used in our study, simply considers the number of cointegrating relationships is at least one. In our study, Johansen's test is used to select pairs for multiple trading under the cointegration approach.

3.5.2 Financial Networks

In this Section we propose stocks selection using the financial networks approach.

Network analysis is a branch of data analysis that uses graph theory and networks as its foundation to look into hidden structures. Network graphs are a great visualization tool which helps to identify the connections between

the objects. Networks are a popular method for representing complex systems because they can straightforwardly express relationships between components and are broadly applicable across various fields. Nodes and edges are the fundamental components of a network graph. Nodes are the objects in the network and edges are the lines connecting them.

A network G= (V, E) is a collection of vertices (nodes) and edges (links) that connect those nodes. For example, in a Twitter social network model, each user is represented as a node, and an edge represents the connection between users who follow each other. Edges in graphs can be directed or undirected depending on whether the connections between nodes (edges) are one-way or two-way. An arrowhead on an edge indicates a direction on a directed graph. One-way relationship is indicated by the beginning and ending nodes on these edges. In Twitter, edges are directed, because people we follow do not necessarily follow us back. In Facebook, edges are undirected as the relationship is mutual or two-way. Therefore, edges of undirected graphs have no arrowheads. Edges can also be weighted or unweighted. For example, in an airline network, each node represents an airport, and edges are weighted based on the distance between two airports. A social network of friendship is an example of unweighted network as, in this case, an edge represents presence or absence of friendship between two individuals (nodes). A network is said to be connected, if a path (edge) connects each pair of nodes in the network.

Network models based on correlation between stock prices were studied by

(a) Directed Network (b) Undirected Network

Figure 3.2: Directed vs Undirected Network Graphs

(Mantegna, 1999). Since Mantegna's work, many researchers used stock price series to compute correlations and construct network graphs (Jothimani et al., 2018). Due to their ease of interpretation and ease of construction, correlation-based networks have been a popular method for inferring financial networks. Nodes and edges of financial network graphs constructed based on correlation, represent individual stocks and correlations based on a threshold level (Chi et al., 2010) respectively. These are undirected and weighted networks, where weights are associated with the strength of the correlation.

Empirical Correlation

Pearson's correlation coefficient between returns or logarithmic returns of stock pairs is commonly used to create stock market network graphs in the literature (Jothimani et al., 2018; Bonanno et al., 2004).

To analyze the relationships between historical stock prices, a correlation

matrix is constructed. This matrix is composed of correlation coefficients ($\rho_{i,j}$) that are calculated for every possible pair of stock prices P_i and P_j

$$\rho_{i,j} = \frac{E(P_i P_j) - E(P_i)E(P_j)}{\sqrt{Var(P_i)}\sqrt{Var(P_j)}}. \tag{3.1}$$

the numerator representing the covariance between P_i and P_j prices. $Var(P_i)$ is defined as $E[(P_i - E[P_i])^2]$.

This empirical formula provides the average correlation for the entire time window. It will change gradually over time and not respond quickly to recent changes, where as a short-term estimate will ignore the historical correlation in favour of what has happened more recently. This idea led us to consider both empirical correlation and Exponentially Weighted Moving Average (EWMA) correlation to construct the financial networks.

Data-driven EWMA (DDEWMA) correlation

We compute DDEWMA correlation for the first differenced series ($d_{i,t}$) of daily prices, instead of considering returns or log-returns series. Specifically, we define

$$d_{i,t} = P_{i,t} - P_{i,t-1}. \tag{3.2}$$

However, using $d_{i,t}$ series makes no difference as can be inferred from Table 3.1 in Section 3.6.3, all the first differenced (I (1)) series are stationary within

the considered time frame.

The DDEWMA correlation matrix is obtained by computing the covariance matrix of the standardised residuals of the $d_{i,t}$ series. The steps are given under Algorithm 4, as described in Zhu et al. (2020).

The time window $[1, T_1]$ is considered as the training period, and $[T_1 + 1, T_2]$ is considered as the testing period. The smoothing constant α, typically set between 0 and 1, determines the weight given to the most recent observations relative to past observations. The optimal value of α is determined by minimizing the Mean Square Error (MSE) between the observed and smoothed volatility of d_t series during the training period. This optimal α is then applied in the testing period to forecast the volatility of d_t series. The computed correlation matrices are symmetric and a value corresponding to a particular row and a column provides the strength of the relationship between the two stocks associated with the specific row and column.

Threshold Correlation

The next step is to find a threshold level to the absolute correlation matrix to filter weaker connections to generate a sparser network. When finding a threshold, the resulting network should be fully connected to apply PageRank, as it is a Markov process, where each node is reachable directly or indirectly via multiple paths. As the threshold gets closer to zero, the network becomes fully connected. Overall, we observe that the total number of node connections decreases with increasing threshold. It is observed that a high enough

Algorithm 4 Data-driven EWMA correlation matrix

Require: $P_t, \quad t = 1, \ldots, k, \ldots, T_1, \ldots, T_2$

$d_t \leftarrow P_t - P_{t-1}, t = 1, \ldots T_2$ First differenced (I(1)) series

$\hat{\rho} \leftarrow Corr(d - mean(d), sign(d - mean(d)))$

$Z_t \leftarrow |d_t - mean(d)| / \hat{\rho}$ Observed volatility

$S_0 \leftarrow \frac{\sum_{t=1}^{k} Z_t}{k}$ Initial volatility forecast

$\alpha \leftarrow (0, 1)$ Set a range for smoothing constant

$S_t \leftarrow \alpha Z_t + (1 - \alpha) S_{t-1}, t = 1, \ldots, T_1$

$\alpha_{opt} \leftarrow min_\alpha \sum_{t=k+1}^{T_1} (Z_t - S_{t-1})^2$ Determine optimal α by minimizing MSE

for t $\leftarrow 1, T_2$ **do**

    $S_t \leftarrow \alpha_{opt} Z_t + (1 - \alpha_{opt}) S_{t-1}$

    $res_t \leftarrow \frac{d_t - mean(d)}{S_t}$

end for

$cor^{dd}[t] \leftarrow Corr(res_t)$

return $cor^{dd}[t]$

threshold with a fully connected graph is particularly interesting because the resulting network would connect stocks that have relatively similar daily price fluctuations. A sequence of values is selected between the 0% - 50% percentiles of correlation values with an increment of 0.01 to identify a threshold with a fully connected graph. Figure 3.3 shows quantiles vs DDEWMA correlation values for the first window of data. A useful threshold is identified as 0.25 for the correlation matrix generated for this interval. Hence, when constructing the network graph, all the correlation coefficients less than 0.25 were set to zero to form a network with minimal complexity. The PageRank algorithm was finally used to rank the importance of the nodes in the network graphs generated for each window, and nodes with adjacent PageRank scores were combined to form appropriate combinations of stocks for pairs trading.

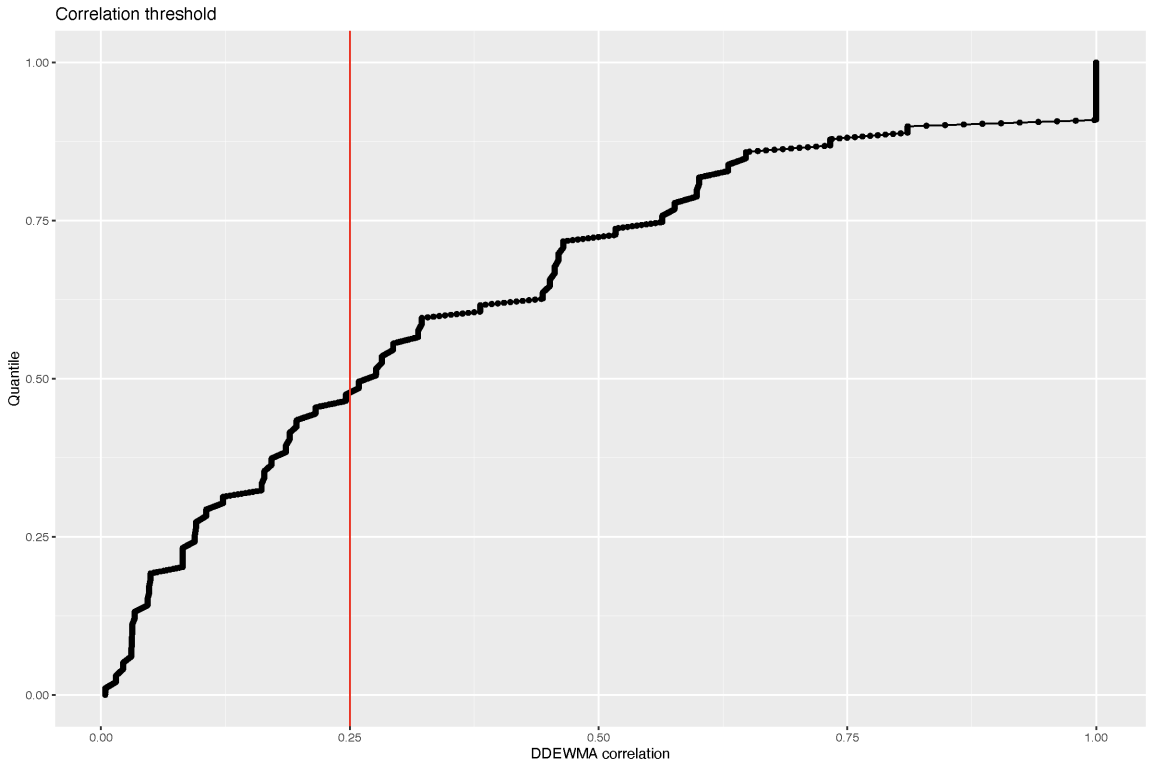


Figure 3.3: Correlation threshold for window 0

The choice of window size also has an impact on the correlation-based network graph's structure. When working with daily data, windows can range from a few months to a few years (Chi et al., 2010). We decided to work with a rolling window of 180 days, which is roughly equal to 9 months as the stock market operates 5 days a week. The trading strategies for the selected rolling windows are reported based on three approaches: (1) cointegrated stocks, (2) dynamic networks and ranks generated using empirical correlation of stock price series, (3) networks and ranks generated using DDEWMA correlation of the $d_{i,t}$ series.

3.5.3 PageRank Algorithm

When we have formed a network, finding the nodes that are most "central" to the network is frequently of interest. Centrality, in networks, is defined in a multitude of ways. In recent years, PageRank has been the most popular algorithm for ranking network nodes. Google is a well-known example that uses PageRank to rank results based on the relevance of the page linked to the search terms. This is widely used in domains such as financial markets, eco-systems and recommendation analytics (Yang et al., 2014). For example, Yang et al. (2014) use PageRank on a directed and weighted network based on cointegration instead of correlation to obtain node importance for 26 global stock market indices. They combine the PageRank algorithm and the cointegration network to identify the most influential stock market index and rank the influence of each index.

When PageRank is applied to a financial network with stochastic price correlations as edges, it can provide insights into the importance or influence of each node within the network. In financial networks, nodes with higher PageRank scores are considered more important and have a more significant impact on the overall network dynamics. They have a greater ability to influence or impact other prices in the network.

PageRank includes a parameter called damping factor (teleport) that decides how fast the algorithm converges. It is a value between 0 and 1, and the higher the value of the damping factor, the higher the chance to follow links from one

node to another rather than jumping randomly. Adjusting the damping factor can influence the sensitivity of the algorithm to the structure of the network. A damping factor in a range of 0 to 1 with an increment of 0.005 is tested and the importance rank of each stock is determined for rolling time windows over the chosen time frame under a damping factor of 0.80. In other words, 80% of the time, the algorithm is assuming that the correlations between stocks are the basis for the relationships (or links) between them. The algorithm does, however, permit random jumps 20% of the time.

Given an initial equal PageRank score of $1/N$ for each node, where N is the total number of nodes in the network, the PageRank score, $PR(j)$ for each node j is updated as:

$$PR(j) = \frac{t}{N} + (1-t) \sum_{k \in M(j)} \frac{PR(k)}{L(k)},$$

where t is set to 0.2. t/N represents the probability of randomly jumping to any node. $(1-t)\sum_{k \in M(j)} \frac{PR(k)}{L(k)}$ represents the probability of following links or correlation relationships, where $M(j)$ is the set of nodes that link to node j, $PR(k)$ is the PageRank of node k and $L(k)$ is the number of outbound links from node k. The algorithm iterates until the PageRank scores converge, meaning the scores between consecutive iterations differ by less than 0.0001.

3.5.4 The spread model

A collection of nonstationary time series is said to be cointegrated if a linear combination of the series is stationary. According to this definition, the spread or difference between two prices, given by

$$\epsilon_t = P_{1,t} - \beta_0 - \beta_1 P_{2,t}\,, \tag{3.3}$$

is stationary if two time series of prices, $P_{1,t}$ and $P_{2,t}$, are cointegrated. This implies that ϵ_t is disturbed around an equilibrium value. The amount of one asset (stock) to buy or sell for every unit of the other asset is indicated by the regression coefficient β_1, also known as the hedge ratio in pairs trading. The hedge ratio varies over time during the trading period, and the trading system is dynamic and updated with new information in real time. Therefore, the linear state space model or dynamic linear model is utilised to incorporate the time-varying regression coefficients in our study (see Section 3.5.7).

3.5.5 State space model and Kalman filtering

A state space model is a mathematical framework for describing and analysing dynamic systems. It is also referred to as a state-space representation or just a state model. Numerous disciplines, including control theory, signal processing, econometrics, and machine learning, heavily rely on state space modelling.

In a state space model, the underlying system is represented by two main components:

1. State equation: The state equation provides an explanation of how the system has changed over time. It symbolises both the impact of outside inputs and the link between the system's prior and current states.
2. Observation equation: It connects the underlying state variables to the system's observed measurements or outputs.

Complex systems with many variables, dynamic behaviour, and uncertainty can be modelled and analysed using a state space model framework. Some key features and applications of state space models are:

- Identification of the system: Based on observed inputs and outputs, state space models can be used to estimate or identify the unknown parameters of a dynamic system. Time series analysis and control systems engineering both make extensive use of the system identification process.
- Filtering and prediction: For tasks involving the estimation of the present or future state of the system from noisy or insufficient observations, state space models can be utilised. State space models serve as the foundation for well-known filtering algorithms like the Kalman filter.
- Time series analysis: For the analysis of time series data and the modelling of intricate temporal dependencies, state space models are extensively utilised. Non-linear dynamics, seasonality, trends, and other time-dependent patterns in the data can all be captured by these models.

A random walk model often serves as the state equation in the state space model (Liang et al., 2020). For instance, we can write

$$\beta_t = \beta_{t-1} + \upsilon_t,\ \upsilon_t \overset{\text{iid}}{\sim} (0, \Sigma_\upsilon)\,, \tag{3.4}$$

where $\beta_t = (\beta_{0,t}, \beta_{1,t}, \ldots, \beta_{m,t})$. β_t is the m-dimensional state vector at time t. Then, an observed process $y_t = P_{1,t}$ can be described by an observation equation as

$$y_t = A_t\beta_t + \epsilon_t,\ \epsilon_t \overset{\text{iid}}{\sim} (0, \sigma^2_\epsilon). \tag{3.5}$$

Here, A_t is a feature matrix of the dimension m, where, $A_t = (1, P_{2,t}, \ldots, P_{m,t})$. The dynamic filtered estimates for the hedge ratio, β_t, given the data $\mathcal{F}^y_t = \{y_1, \ldots, y_t\}$, is given by, $\widehat{\beta}_{t|t} = E[\beta_t|\mathcal{F}^y_t]$.

Data-Driven Maximum Informative Filters Using Estimating Functions

For the state space and observation models, (Equation 3.4 and Equation 3.5), let $\widehat{\beta}_{t-1|t-1} = E[\beta_{t-1}|\mathcal{F}^y_{t-1}]$ and $I^{-1}_{t-1|t-1} = Var(\beta_{t-1} - \widehat{\beta}_{t-1|t-1}|\mathcal{F}^y_{t-1})$. Then using non-Gaussian maximum informative filter provided in Thavaneswaran and Thompson (2019), two elementary estimating functions (g_{1t} and ν_t) are combined as follows,

$$g_{1t} = \beta_t - \widehat{\beta}_{t-1|t-1}, \quad \nu_t = y_t - A_t\widehat{\beta}_{t-1|t-1}\,, \tag{3.6}$$

where $\nu_t = y_t - \hat{y}_{t|t-1}$ is the innovation or forecast error of y_t. The optimal linear combination in the class of linear combinations of g_{1t} and ν_t is given by the estimating function,

$$\beta_t - \widehat{\beta}_{t-1|t-1} - \frac{Cov(g_{1t}, \nu_t|\mathcal{F}^y_{t-1})}{Var(\nu_t|\mathcal{F}^y_{t-1})}(y_t - A_t\widehat{\beta}_{t-1|t-1})\,.$$

This yields the optimal estimate of β_t as,

$$\widehat{\beta}_t = \widehat{\beta}_{t-1|t-1} + (I^{-1}_{t-1|t-1} + \Sigma_\upsilon)A'_t Q^{-1}_t (y_t - A_t\widehat{\beta}_{t-1|t-1}), \tag{3.7}$$

where Q_t is the innovation variance given by

$$Q_t = A_t(I^{-1}_{t-1|t-1} + \Sigma_\upsilon)A'_t + \sigma^2_\epsilon\,.$$

The linear optimal filter (Equation 3.7) is known as the Kalman filter (KF) when Gaussian assumptions are met for υ_t and ϵ_t. Using Kalman filtering for parameter estimation is one approach to include both dynamism and uncertainty in our decisions. KF provides a computationally effective recursive process for dynamic learning, enabling the acquisition of dynamical systems by the use of preexisting knowledge. The use of KF in multiple trading and pairs trading has gained popularity recently. The recursive form of the non-Gaussian maximum informative filter algorithm was proposed in Liang et al. (2020a) as an extension for the standard KF algorithm. In this study, we are introducing a newer version of the algorithms in Liang et al. (2020a), which can provide

optimal trading signals and profits for any combination of pairs and multiple trading strategies within data windows.

3.5.6 Estimating Innovation volatility

There are two methods of estimating innovation volatility in the literature. The commonly used method is the square root estimator of the KF innovation variance Q_t, ($\sqrt{Q_t}$ as the innovation volatility forecast). However, this square root estimator is not an appropriate estimator in the current context as the assumption of normality required for it to be a sufficient statistic is violated for financial data. Moreover, trading profits get affected by initial values when $\sqrt{Q_t}$ is used (Longmore, 2019; Liang et al., 2020a). Hence, a more robust volatility forecasting model is proposed in Thavaneswaran et al. (2020) to forecast the volatility of innovation series, ν_t as defined in Equation 3.6.

The DDEWMA volatility forecasting model presented as Equation 3.8 is used to obtain data-driven innovation volatility forecasts (DDIVF). This is the second method that we use to obtain volatility estimates of innovation variance. Let σ_t^2 is the conditional variance of innovation ν_t, depending on prior data up until time $t-1$. Then, we can estimate σ_t using

$$\hat{\sigma}_t = (1-\alpha)\,\hat{\sigma}_{t-1} + \alpha\frac{|\nu_{t-1}-\bar{\nu}|}{\hat{\rho}_\nu}, \quad 0<\alpha<1, \tag{3.8}$$

where α is the smoothing constant and $\hat{\rho}_\nu$ is the sample sign correlation of the innovation sequence given by

Algorithm 5 Dynamic DD-EWMA volatility forecasts of innovation

Require: Predicted errors $\nu_s, s = t-k, \cdots, t-1$

1: $\hat{\rho}_\nu \leftarrow \text{Corr}(\nu_s - \bar{\nu}, \text{sign}(\nu_s - \bar{\nu}))$
2: $V_s \leftarrow |\nu_s - \bar{\nu}|/\hat{\rho}_\nu$ {Compute estimated volatility}
3: $S_{t-k-1} \leftarrow \bar{V}_l$ {Initial volatility forecast using first l observations}
4: $\alpha \leftarrow (0,1)$ *by* 0.01{Set a range for α}
5: $S_s \leftarrow \alpha * V_s + (1-\alpha) * S_{s-1}, s = t-k, \ldots, t-1$
6: $\alpha_{opt} \leftarrow \min_\alpha \sum_{s=t-k+l}^{t-1} (V_s - S_{s-1})^2$ {Determine optimal α by minimizing FESS}
7: $S_s \leftarrow \alpha_{opt} * V_s + (1-\alpha_{opt}) * S_{s-1}, s = t-k, \ldots, t-1$
8: $\hat{\sigma}_t \leftarrow S_{t-1}$ {Calculate one-step-ahead DDIVF based on k observations}
9: **return** $\alpha_{opt}, \hat{\sigma}_t$

$$\text{Corr}(\nu_t - \bar{\nu}, \text{sign}(\nu_t - \bar{\nu}))\,.$$

In Equation 3.8, the conditional distribution of ν_t is identified by the sample sign correlation $\hat{\rho}_\nu$ and the optimal value of α (α_{opt}) is obtained by minimising the one-step ahead forecast error sum of squares (FESS). This makes the Model given in Equation 3.8 data-driven. Algorithm 5 illustrates the details of DDIVF calculation.

The sample sign correlation $\hat{\rho}_\nu$ and volatility estimate $|\nu_s - \overline{\nu}|/\hat{\rho}_\nu$ are computed based on the past k innovations $\nu_{t-k}, \ldots, \nu_{t-1}$. The smoothed value S_s of the volatility estimate is computed recursively using the optimal value α_{opt}. Finally, the computed S_{t-1} is used as the volatility forecast $\hat{\sigma}_t$ for innovation ν_t.

The innovation sequence ν_t at time t, and its standard error $\sqrt{Q_t}$ or DDIVF (more robust estimator) are used to generate trading signals and generate

Algorithm 6 Estimating hedge ratios using Non-Gaussian maximum informative filtering

Require: Adjusted closing prices data $P_{1,t}, P_{2,t}, \ldots, P_{m,t}$ where, $t = 1, \ldots, n$

Let $y_t = P_{1,t}$ $A_t = (1, P_{2,t}, \ldots, P_{m,t})$
Initial values: $\boldsymbol{\beta}_0$, I_0, Σ_υ, σ_ϵ^2
for $t \leftarrow 1, \ldots, n$ **do**
    Prediction: Based on data available at $t-1$:
    $\widehat{\boldsymbol{\beta}}_{t|t-1} \leftarrow \widehat{\boldsymbol{\beta}}_{t-1|t-1}$; $I_{t|t-1}^{-1} \leftarrow I_{t-1|t-1}^{-1} + \Sigma_\upsilon$; $\hat{y}_{t|t-1} \leftarrow A_t \widehat{\boldsymbol{\beta}}_{t|t-1}$
    Update: Inference about the hedge ratio, $\boldsymbol{\beta}_t$ is updated using the observation y_t at time t
    $\nu_t \leftarrow y_t - \hat{y}_{t|t-1}$; $Q_t \leftarrow A_t I_{t|t-1}^{-1} A_t' + \sigma_\epsilon^2$
    DDIVF $\hat{\sigma}_t$ is calculated based on $\nu_{t-k}, \ldots, \nu_{t-1}$ using Algorithm 5
    $\widehat{\boldsymbol{\beta}}_{t|t} \leftarrow \widehat{\boldsymbol{\beta}}_{t|t-1} + I_{t|t-1}^{-1} A_t' Q_t^{-1} \nu_t$; $I_{t|t} = I_{t|t-1} + \frac{1}{\sigma_\epsilon^2} A_t' A_t$
end for
return ν_t, Q_t, $\hat{\sigma}_t$

optimal profits following the steps of Algorithm 7. This study compares the profits generated under different trading strategies when using each estimator of innovation volatility.

3.5.7 Estimating Hedge Ratios

Algorithm 6 provides the pseudo code for estimating the dynamic hedge ratio β_t, which uses the non-Gaussian maximum informative filter, provided appropriate initial values for Σ_υ (covariance of state) and σ_ϵ^2 (covariance of observation).

The innovation and prediction error are approximately normally distributed when there are no outliers (Liang et al., 2020a). Hence, the z-score,

$$z_t = \nu_t / \hat{\sigma}_t \quad \text{or} \quad z_t = \nu_t / \sqrt{Q_t} \tag{3.9}$$

is also following an approximate normal distribution.

This z-score is compared with a threshold value p and trading signals are obtained according to the conditions given under Algorithm 7.

3.5.8 Generating Optimal Profits

Algorithm 7 creates the trading signals s_t, where sells are represented as $s_t = -1$, buys as $s_t = 1$, and no action as $s_t = 0$. The buy signal is generated when z_t crosses p from above; the sell signal is generated when z_t crosses over p from below. Trading positions (buy/sell/hold) are determined using s_t and profit of holding those positions are further calculated using the Algorithm 7. Each trade consists of 1000 units of the spread when trading signals are trigged. For example, the position of A_t corresponding to a sell signal is computed as $-1000 * \widehat{\boldsymbol{\beta}}_{t-1|t-1} * s_{t-1}$. Finally, Annualized Sharpe Ratios (ASR) are computed (step 13 of Algorithm 7) for a range of values of p. The optimal threshold value, p_{opt}, is determined by maximizing the ASR, as this indicates a higher risk-adjusted return for the investment. The estimated cumulative profit is the sum of the price differences multiplied by the corresponding positions in each asset. The trading algorithms possess the ability to consistently evaluate price data, compute spreads, produce trading signals, and carry out trades using pairs trading methods.

Algorithm 7 Cumulative profit using optimal signals

Require: Data: $P_{1,t}, P_{2,t}, \ldots, P_{m,t}, t = 1, \ldots, n; \nu_t, Q_t$ and $\widehat{\boldsymbol{\beta}}_{t-1|t-1}$ obtained from Algorithm 6

Let $y_t = P_{1,t}, A_t = (1, P_{2,t}, \ldots, P_{m,t})$ and $z_t = \nu_t/\sigma_t$ (DDIVF) $z_t = \nu_t/\sqrt{Q_t}$ (Sqrt Q method)

Generate trading signals s_t:

for $t \leftarrow 1, \ldots, n$ **do**

  If $z_{t-1} < p$ & $z_t > p$, then $s_t \leftarrow -1$

  If $z_{t-1} > -p$ & $z_t < -p$, then $s_t \leftarrow 1$

  Else $s_t \leftarrow 0$

  $position.A_t \leftarrow -1000 * \widehat{\boldsymbol{\beta}}_{t-1|t-1} * s_{t-1}$

  $position.y_t \leftarrow 1000 * s_{t-1}$

  $profit.A_t \leftarrow (A_t - A_{t-1}) * position.A_t$

  $profit.y_t \leftarrow (y_t - y_{t-1}) * position.y_t$

  $profit_t \leftarrow profit.A_t + profit.y_t$

end for

Calculate the ASR as $SR(p) = \sqrt{252} * mean(profit_t)/sd(profit_t)$

Determine the optimal value of p, p_{opt} , by maximizing $SR(p)$

Obtain the cumulative profit $cumsum(profit_t)$ using p_{opt}

return p_{opt}, cumulative profit

3.6 Data Application

The techniques and algorithms described in Section 3.5 were tested for pairs and multiple trading between adjusted closing prices of ten assets, S&P 500 index (SPY), Adobe Inc (ADBE), eBay Inc (EBAY), Microsoft Corporation (MSFT), International Business Machines Corporation (IBM), SPDR Gold Trust (GLD), iShares MSCI Australia ETF (EWA) including investment results of an index consisting of Australian equities, iShares North American Natural Resources ETF (IGE) including investment results of an index consisting of equities in the natural resources sector in North America, iShares MSCI Canada

ETF (EWC) including investment results of an index consisting of Canadian equities and VanEck Vectors Gold Miners ETF (GDX) (Finance, 2020). An exchange-traded fund (ETF) is a financial instrument that functions similarly to a stock but tracks an index, commodity, or basket of assets. When considering pairs trading, ETFs offer a distinct advantage over individual stocks: once cointegrated, ETF pairs tend to maintain their relationship more consistently over time. This stability is primarily due to the fact that ETFs represent a diversified basket of assets, and the fundamental economic factors affecting a group of stocks typically evolve more slowly than those impacting a single company (Mota, 2023). As an additional feature to complement the analysis, stocks will be included in the basket of financial assets. Required data were downloaded from Yahoo! Finance for the period from 2017-02-01 to 2023-12-08.

Results are reported under three major periods; (i) pre-COVID, (ii) during COVID and (iii) post-COVID on a rolling window basis as the trend and volatility of assets behave differently during these periods. The ratio between the cumulative profit and the number of transactions that took place within the window is provided as a proxy to the transaction costs incurred in the process of trading. All price series were internally transformed using a base-10 logarithmic function (log_{10}) to enhance the visualizations of Figures 3.4 to 3.6.

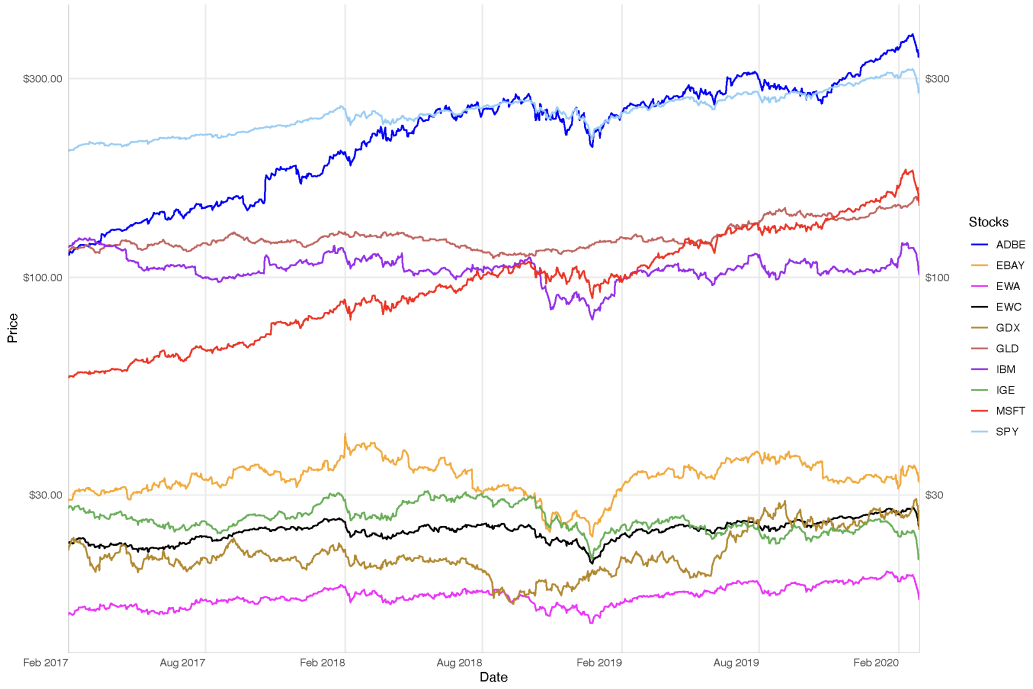


Figure 3.4: Daily Adjusted Closing Prices (USD) during the Pre-Covid period

3.6.1 Pre-COVID period

Figure 3.4 shows the behaviour of price series for the 10 selected stocks during the pre-COVID period from 2017-02-01 to 2020-02-29. This time-frame encompasses data windows 0 to 6. During this period, prices of ADBE and SPY show similar patterns of upward and downward trends. We can observe a downfall in prices during February 2019 except for GLD and GDX.

3.6.2 During COVID period

Figure 3.5 shows the behaviour of price series for the 10 selected stocks during the COVID period, which extends from 2020-03-01 to 2022-02-28 and includes

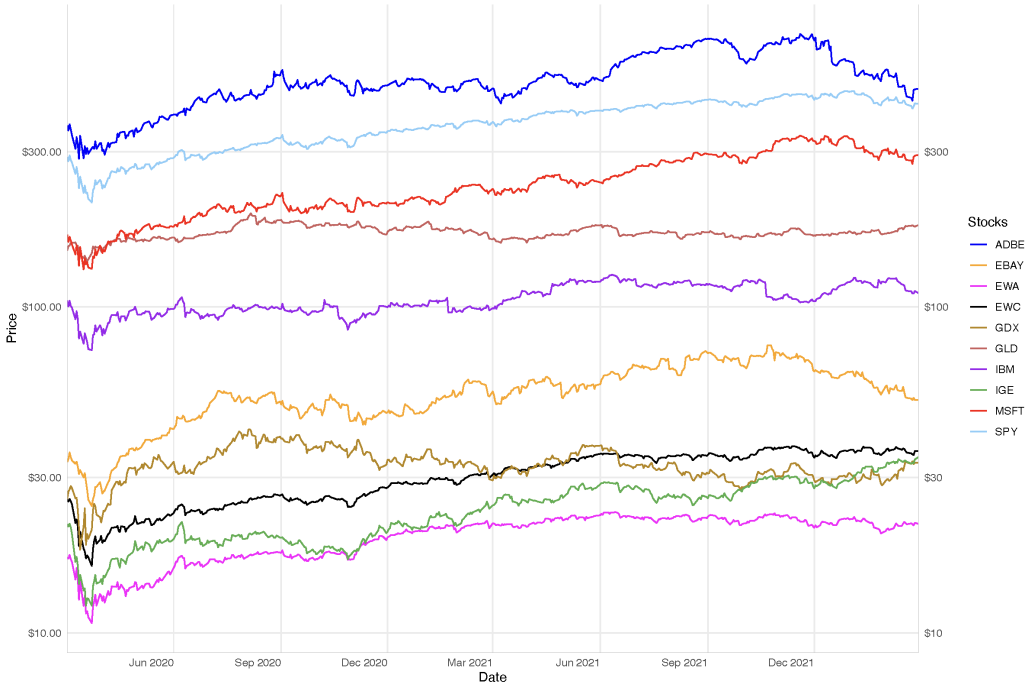


Figure 3.5: Daily Adjusted Closing Prices (USD) during the Covid period

data windows 7 to 12. Throughout this period, all price series exhibited similar patterns, characterized by a common initial decline in prices.

3.6.3 Post-COVID period

Figure 3.6 shows the behaviour of price series for the 10 selected stocks during the post-COVID period, which extends from 2022-03-01 to 2023-12-08 and includes data windows 13 to 17. In the post-COVID period, the selected stocks exhibited relatively stable price patterns, with ADBE being the notable exception. We can observe a downward trend for ADBE until it starts to rise again from June 2023.

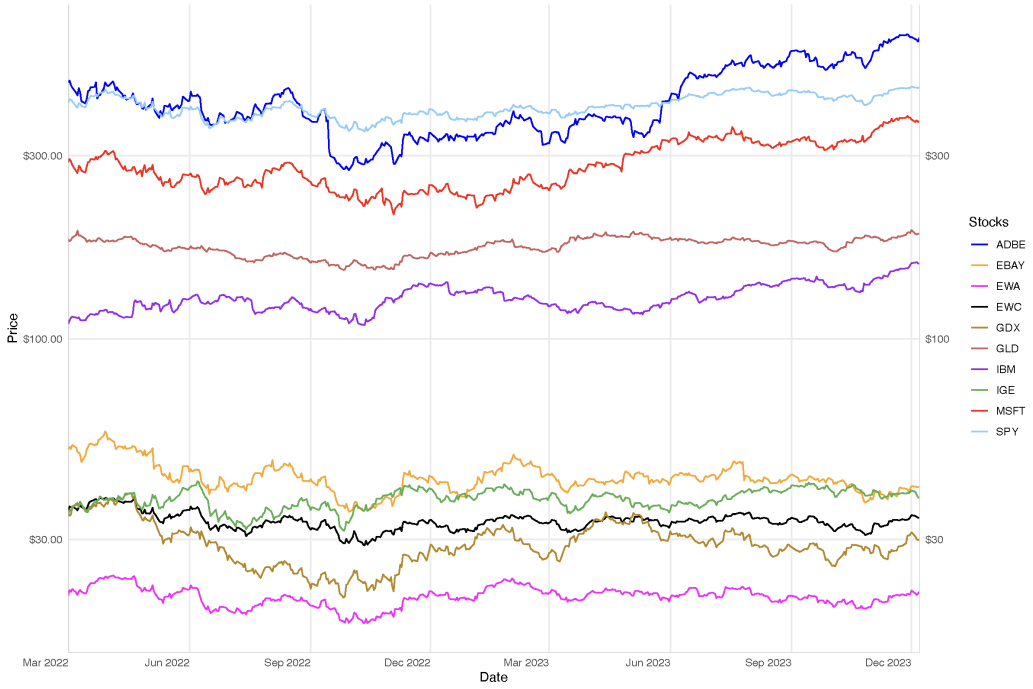


Figure 3.6: Daily Adjusted Closing Prices (USD) during the Post-Covid period

Table 3.1 represents the ARIMA models estimated from R package `fabletools` (O'Hara-Wild et al., 2024) for stocks in each window selected from the three periods. The first row presents the ARIMA models for the selected windows from each period for the stock SPY. Likewise, each row of Table 3.1 presents the relevant ARIMA models for each considered stock. Since the degree of differencing (d) in all the ARIMA models is either 0 or 1, all the first differenced series or the original series itself are stationary. Hence, in our work, we are using the price series and its first differences series (Equation 3.2) to calculate correlation matrices.

In order to obtain realistic profits, it is necessary to apply pairs trading

Table 3.1: ARIMA models for stock prices

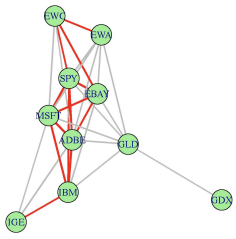
	Pre-COVID		COVID		Post-COVID	
Stock	Window 2	Window 4	Window 8	Window 10	Window 14	Window 16
SPY	ARIMA(3,1,0)	ARIMA(0,1,0)	ARIMA(0,1,2)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,1,2)
ADBE	ARIMA(2,1,0)	ARIMA(0,1,1)	ARIMA(1,1,0)	ARIMA(1,0,0)	ARIMA(1,1,0)	ARIMA(0,1,1)
EBAY	ARIMA(0,1,0)	ARIMA(1,0,1)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,1,0)	ARIMA(0,1,0)
MSFT	ARIMA(2,1,2)	ARIMA(1,0,1)	ARIMA(1,1,0)	ARIMA(0,1,1)	ARIMA(1,1,0)	ARIMA(0,1,2)
IBM	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(2,1,3)	ARIMA(0,1,0)	ARIMA(0,1,1)	ARIMA(1,1,1)
GLD	ARIMA(0,1,0)	ARIMA(0,1,1)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,1,0)
GDX	ARIMA(1,0,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,1,0)
EWA	ARIMA(1,0,3)	ARIMA(2,1,1)	ARIMA(0,1,2)	ARIMA(0,1,1)	ARIMA(0,1,0)	ARIMA(1,1,0)
EWC	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(2,1,2)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,1,0)
IGE	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(0,1,0)	ARIMA(1,0,0)	ARIMA(1,1,0)

for cointegrated or correlated assets in a previous window. Hence, for the pre-COVID period analysis, we utilized data from windows 0 and 1 to establish trading pairs and applied on data from windows 2 and 4 respectively to compute profits. Data for window 2 covered the period from 2017-10-18 to 2018-07-06 and data for window 4 covered the period from 2018-07-09 to 2019-03-26. In our analysis for COVID period, pairs were established using data from windows 7 and 8 and applied in windows 8 and 10 respectively to compute profits. Data for windows 8 and 10 span from 2019-12-11 to 2020-08-27 and from 2020-08-28 to 2021-05-17 respectively. For the post-COVID period analysis, we utilized data from windows 13 and 14 to establish pairs and applied in windows 14 and 16 respectively to compute profits. Data for windows 14 and 16 span from 2022-02-02 to 2022-10-19 and from 2022-10-20 to 2023-07-11 respectively.

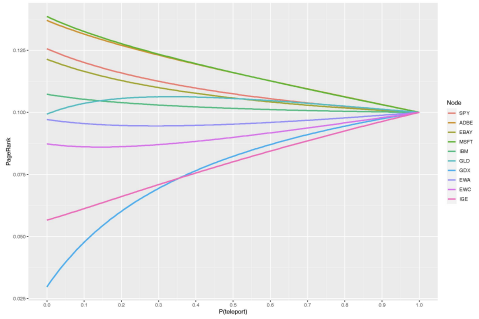
3.6.4 Financial Network Graphs

Pre-COVID

Figures 3.7 to 3.12 show financial networks and their PageRank plots, which demostrate how the PageRank score changes with the damping factor using price series (P_t) and differenced price (d_t) series separately. The plots present the three major periods separately, focusing on the windows used to establish pairs within each period. Observe that all the networks are fully connected and edges highlighted with red connect highly correlated stocks with a threshold of 0.70.

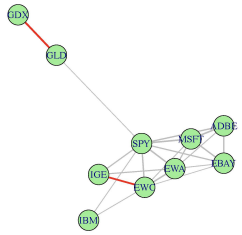


(a) Financial network of window 0

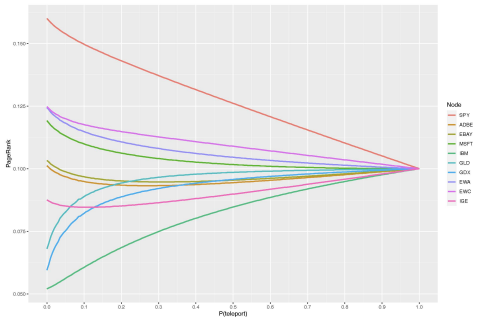


(b) Effect of damping factor on PageRank

Figure 3.7: Dynamic financial network and PageRank plot for window 0 using price series



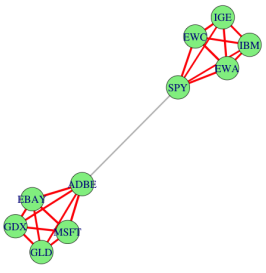
(a) Financial network of window 0



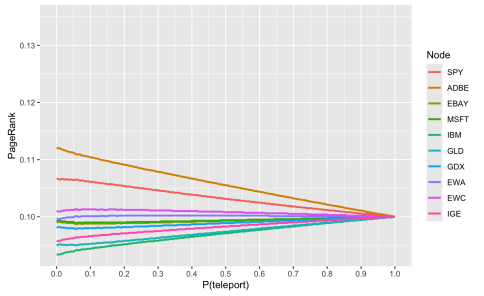
(b) Effect of damping factor on PageRank

Figure 3.8: Dynamic financial network and PageRank plot for window 0 using differenced price series

During-COVID

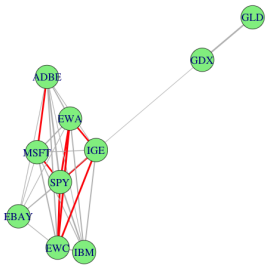


(a) Financial network of window 8

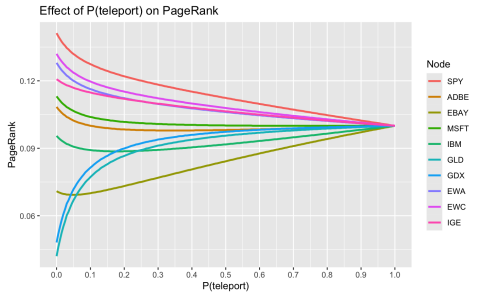


(b) Effect of damping factor on PageRank

Figure 3.9: Dynamic financial network and PageRank plot for window 8 using price series



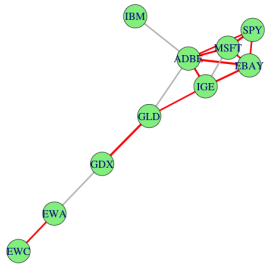
(a) Financial network of window 8



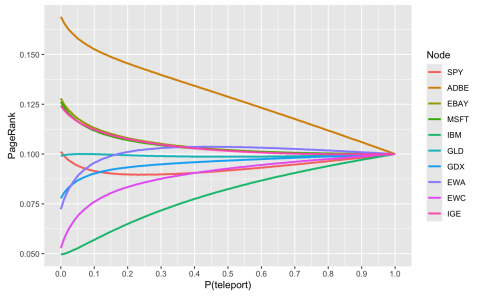
(b) Effect of damping factor on PageRank

Figure 3.10: Dynamic financial network and PageRank plot for window 8 using differenced price series

Post-COVID

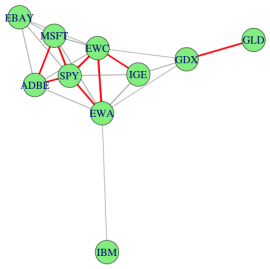


(a) Financial network of window 13

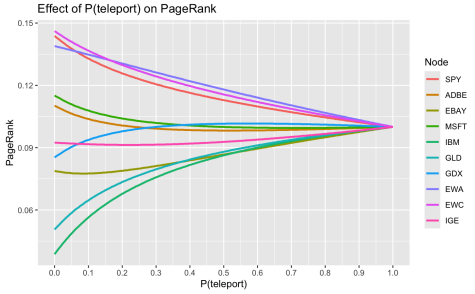


(b) Effect of damping factor on PageRank

Figure 3.11: Dynamic financial network and PageRank plot for window 13 using price series



(a) Financial network of window 13



(b) Effect of damping factor on PageRank

Figure 3.12: Dynamic financial network and PageRank plot for window 13 using differenced price series

3.6.5 Pairs trading with cointegration approach

Table 3.2 compares two trading methods, DDIVF method and the $\sqrt{Q_t}$ method, using various stock combinations formed based on the cointegration approach. Trading results are shown using the DDIVF method and the $\sqrt{Q_t}$ method separately. The column labeled PP Sig (profit per signal) gives the ratio between cumulative profit and the number of transactions that occurred under each trading strategy. Multiple trading strategies with three stocks were formed by combining pairs trading strategies that provided the highest per-transaction profits in the particular window. Multiple trading provides better profits with the DDIVF method compared to the $\sqrt{Q_t}$ method. Pairs trading performs better than multiple trading in windows relevant to pre-COVID and post-COVID periods. During the COVID period (windows 8 and 10), multiple trading strategies provide better profits.

Table 3.2: Rolling window per transaction profits for trading strategies formed using the cointegration approach

Trading Strategy		DDIVF method				$\sqrt{Q_t}$ method			
		ASR	Cum. Profit	TRAN	PP Sig	ASR	Cum. Profit	TRAN	PP Sig
	Pre-Covid								
	Window 2								
PT	EBAY/SPY	1.65	20227.31	10	2022.73	-0.69	-21859.03	57	-383.49
	MSFT/SPY	2.80	32589.11	8	4073.64	3.72	55456.28	44	1260.37
	EBAY/MSFT	1.99	22589.97	15	1506.00	3.66	42502.22	60	708.37
	EWC/EBAY	2.15	6625.53	21	315.50	0.56	1749.20	24	72.88
MT	MSFT/EBAY/SPY	3.13	29899.23	10	2989.92	2.02	29407.27	52	565.52
	Window 4								
PT	EBAY/SPY	2.80	59433.32	11	5403.03	-1.02	-27711.62	56	-494.85
	MSFT/SPY	3.90	76339.58	35	2181.13	3.37	66864.29	61	1096.14
	EBAY/MSFT	1.85	24509.42	25	980.38	2.77	38249.82	53	721.69
	EWC/EBAY	1.04	2780.19	28	99.29	0.94	3088.06	25	123.52
MT	MSFT/EBAY/SPY	4.19	72749.43	23	3163.02	3.41	61113.99	58	1053.69
	During Covid								
	Window 8								
PT	ADBE/SPY	2.96	112729.32	30	3757.64	2.35	101065.40	45	2245.90
	IBM/ADBE	1.45	66800.69	5	13360.14	-3.13	-236221.16	48	-4921.27
	EWC/IBM	2.34	23759.89	25	950.40	1.64	19563.91	60	326.07
	EWC/ADBE	1.25	76901.60	38	2023.73	-0.23	-17483.83	56	-312.21
	IBM/SPY	0.57	9854.10	18	547.45	-0.54	-13039.26	46	-283.46
MT	IBM/ADBE/SPY	3.02	84078.93	27	3114.03	2.71	86856.29	42	2068.01
	EWC/IBM/ADBE	2.10	112572.87	6	18762.15	-2.88	-212261.00	48	-4422.10
	EWC/ADBE/SPY	3.67	135549.58	29	4674.12	2.35	98782.08	45	2195.16
	Window 10								
PT	GDX/EBAY	4.94	44145.78	36	1226.27	1.13	11437.33	58	197.20
	GDX/MSFT	0.68	24065.43	40	601.64	-0.35	-12844.15	58	-221.45
MT	GDX/EBAY/MSFT	2.86	80665.98	16	5041.62	1.57	48800.53	60	813.34
	Post-Covid								
	Window 14								
PT	EWC/MSFT	1.88	36871.44	9	4096.83	1.40	39253.70	54	726.92
	IGE/EWC	2.23	3707.74	9	411.97	0.87	2673.81	19	140.73
	EBAY/MSFT	1.78	52673.74	5	10534.75	0.71	25692.44	61	421.19
MT	EWC/EBAY/MSFT	2.14	28537.98	14	2038.43	1.04	26810.37	62	432.43
	Window 16								
PT	EWA/SPY	1.23	28696.29	44	652.19	0.79	19071.90	66	288.97
	EBAY/MSFT	0.51	21247.32	18	1180.41	-0.86	-37304.86	53	-703.87
	IGE/MSFT	2.59	105941.32	28	3783.62	1.18	54168.54	63	859.82
	EWC/EBAY	2.07	9379.48	30	312.65	1.08	4944.06	22	224.73
	EWA/EWC	1.79	2437.73	35	69.65	1.75	2379.80	21	113.32
MT	EWC/EBAY/MSFT	0.77	26677.70	17	1569.28	0.27	10085.33	55	183.37

PT - Pairs Trading, MT - Multiple Trading
TRAN - Number of transactions, PP Sig - Profit per signal,
ASR - Annualized Sharpe ratio, DDIVF - Data-driven innovation volatility forecast

3.6.6 Pairs trading with empirical correlation-based financial networks

Table 3.3 compares the DDIVF and the $\sqrt{Q_t}$ methods with stock combinations formed based on the rankings obtained by the dynamic financial network generated using the historical price series values. In this approach also, the DDIVF method performs better than the $\sqrt{Q_t}$ method in terms of their per-transaction profits. Moreover, both pairs trading and multiple trading perform equally well in this approach.

Table 3.3: Rolling window per transaction profits for trading strategies formed using the price series (P_t)

Trading Strategy		DDIVF method				$\sqrt{Q_t}$ method			
		ASR	Cum. Profit	TRAN	PP Sig	ASR	Cum. Profit	TRAN	PP Sig
	Pre-covid								
	Window 2								
PT	MSFT/ADBE	1.11	17631.03	39	452.08	0.87	13979.54	58	241.03
	ADBE/SPY	5.30	58093.92	26	2234.38	4.70	51976.93	22	2362.59
	EBAY/SPY	1.65	20227.43	10	2022.74	-0.69	-21858.26	57	-383.48
	EBAY/IBM	2.72	21554.71	10	2155.47	-0.36	-5897.48	45	-131.06
MT	EBAY/ADBE/SPY	4.91	49988.70	27	1851.43	4.31	44249.34	25	1769.97
	Window 4								
PT	MSFT/ADBE	2.85	62733.81	9	6970.42	3.66	89983.33	50	1799.67
	ADBE/SPY	4.62	137465.98	13	10574.31	4.21	128022.96	29	4414.58
	EBAY/SPY	2.80	59433.54	11	5403.05	-1.02	-27712.90	56	-494.87
	EBAY/IBM	3.50	45606.55	14	3257.61	2.56	34343.56	50	686.87
MT	MSFT/ADBE/SPY	4.08	104330.45	25	4173.22	4.74	120751.46	28	4312.55
	During Covid								
	Window 8								
PT	EWC/SPY	3.51	32985.62	11	2998.69	0.22	5153.46	61	84.48
	EWA/SPY	3.54	106282.66	37	2872.50	4.42	134227.78	72	1864.27
	EWA/ADBE	2.18	141755.53	10	14175.55	-0.01	-911.31	53	-17.19
	GDX/MSFT	1.77	79829.62	40	1995.74	1.99	80617.26	60	1343.62
	GDX/GLD	4.96	77471.60	20	3873.58	0.85	30438.89	60	507.31
	GLD/MSFT	5.00	65252.16	10	6525.22	3.72	122545.58	62	1976.54
MT	EWA/MSFT/ADBE	2.92	129757.85	10	12975.78	2.01	99620.19	45	2213.78
	EWA/GLD/ADBE	3.66	136785.03	12	11398.75	3.01	227675.79	63	3613.90
	Window 10								
PT	EWA/IGE	3.36	6681.63	7	954.52	1.61	3262.69	20	163.13
	EBAY/MSFT	2.88	104851.12	34	3083.86	1.49	57390.50	62	925.65
	GDX/MSFT	0.68	24065.26	40	601.63	-0.35	-12844.41	58	-221.46
	GLD/ADBE	1.76	83887.75	8	10485.97	0.64	37571.58	61	615.93
MT	GLD/MSFT/ADBE	3.62	81413.97	7	11630.57	1.64	64456.67	35	1841.62
	GLD/EBAY/ADBE	2.46	112512.76	13	8654.83	0.31	17545.52	63	278.50
	Post-Covid								
	Window 14								
PT	EWC/SPY	1.30	25029.77	17	1472.34	-0.98	-25155.58	54	-465.84
	EWC/MSFT	1.88	36871.49	9	4096.83	1.40	39252.78	54	726.90
	EWA/MSFT	2.01	41548.87	23	1806.47	1.59	40575.67	58	699.58
	GDX/ADBE	2.69	196724.16	11	17884.01	1.12	96621.49	62	1558.41
MT	GDX/EWC/ADBE	2.38	152309.03	20	7615.45	-0.05	-3299.87	60	-55.00
	Window 16								
PT	EBAY/SPY	2.20	83109.95	30	2770.33	0.97	37239.35	59	631.18
	MSFT/SPY	2.91	75877.37	19	3993.55	2.13	78708.07	25	3148.32
	GLD/MSFT	2.47	95488.41	35	2728.24	1.82	72901.10	53	1375.49
	IGE/IBM	3.14	34373.33	21	1636.83	2.05	23446.84	57	411.35
MT	EBAY/MSFT/SPY	3.05	73601.09	18	4088.95	2.52	86060.15	18	4781.12

PT - Pairs Trading, MT - Multiple Trading

TRAN - Number of transactions, PP Sig - Profit per signal,

ASR - Annualized Sharpe ratio, DDIVF - Data-driven innovation volatility forecast

3.6.7 Pairs trading with DDEWMA correlation-based financial networks

Table 3.4 compares the DDIVF and the $\sqrt{Q_t}$ trading methods with stock combinations formed based on the rankings obtained by the financial network generated using $d_{i,t}$ series. In this approach as well, the DDIVF method provides better per-transaction profits than the $\sqrt{Q_t}$ method. Here, pairs trading strategies formed using financial network and node rankings give higher per transaction profits than multiple trading strategies.

Table 3.4: Rolling window per transaction profits for trading strategies formed using the differenced series (d_t)

Trading Strategy		DDIVF method				$\sqrt{Q_t}$ method			
		ASR	Cum. Profit	TRAN	PP Sig	ASR	Cum. Profit	TRAN	PP Sig
	Pre-covid								
	Window 2								
PT	MSFT/SPY	2.80	32588.83	8	4073.60	3.71	55455.84	44	1260.36
	IGE/ADBE	4.30	76166.66	22	3462.12	1.80	47390.80	66	718.04
	GDX/GLD	1.71	10365.69	9	1151.74	4.79	35640.66	70	509.15
	EBAY/GLD	1.28	16591.09	28	592.54	1.39	25601.08	61	419.69
	EBAY/IBM	2.72	21554.70	10	2155.47	-0.36	-5897.33	45	-131.05
MT	MSFT/SPY/ADBE	5.61	55238.98	24	2301.62	5.61	55238.98	20	2761.95
	IGE/MSFT/SPY	3.46	31669.82	9	3518.87	3.29	43720.31	60	728.67
	Window 4								
PT	MSFT/SPY	3.90	76339.76	35	2181.14	3.37	66864.84	61	1096.14
	EWA/MSFT	3.61	38415.68	33	1164.11	2.72	29345.39	55	533.55
	IGE/ADBE	4.21	154763.37	28	5527.26	1.64	62239.01	56	1111.41
	GDX/GLD	2.76	30172.01	30	1005.73	1.72	19095.41	60	318.26
	EBAY/IBM	3.50	45606.57	14	3257.61	2.56	34343.36	50	686.87
MT	ADBE/IGE/SPY	4.66	132728.52	14	9480.61	4.26	123664.75	23	5376.73
	IBM/IGE/SPY	0.67	12242.73	5	2448.55	2.55	55310.77	45	1229.13
	During Covid								
	Window 8								
PT	EWC/SPY	3.51	32985.82	11	2998.71	0.22	5153.60	61	84.49
	EWC/MSFT	1.07	24002.85	16	1500.18	-0.09	-2282.08	56	-40.75
	EWA/GLD	2.34	87540.01	38	2303.68	3.48	128500.83	66	1946.98
	GDX/GLD	4.96	77471.84	20	3873.59	0.85	30439.08	60	507.32
	GDX/ADBE	1.61	162163.92	8	20270.49	0.61	56275.70	56	1004.92
MT	GDX/ADBE/SPY	4.05	148806.82	31	4800.22	2.11	89201.52	43	2074.45
	GDX/EWC/ADBE	1.35	93601.04	39	2400.03	0.15	9833.10	52	189.10
	Window 10								
PT	EWA/SPY	1.96	34618.38	26	1331.48	1.06	20519.16	61	336.38
	IBM/MSFT	1.55	35731.55	7	5104.51	2.71	96262.52	62	1552.62
	GDX/GLD	2.59	38680.26	24	1611.68	1.81	29884.54	66	452.80
	ADBE/EBAY	1.88	138440.63	31	4465.83	0.74	62276.53	60	1037.94
MT	IBM/MSFT/ADBE	1.75	39821.81	12	3318.48	0.54	21193.50	51	415.56
	IBM/EBAY/MSFT	1.97	52660.00	27	1950.37	3.96	115333.45	59	1954.80
	Post-Covid								
	Window 14								
PT	EWC/SPY	1.30	25029.39	17	1472.32	-0.98	-25155.08	54	-465.83
	EWA/MSFT	2.01	41549.01	23	1806.48	1.59	40576.34	58	699.59
	IGE/MSFT	1.79	72695.84	27	2692.44	-2.63	-110107.26	49	-2247.09
	IGE/ADBE	4.68	314914.91	10	31491.49	-2.38	-193534.57	49	-3949.69
	GDX/GLD	2.08	41052.09	27	1520.45	1.37	29907.29	64	467.30
MT	IGE/EWA/ADBE	4.65	239539.34	9	26615.48	-0.45	-32215.24	53	-607.83
	Window 16								
PT	MSFT/SPY	2.91	75877.18	19	3993.54	2.13	78708.16	25	3148.33
	IGE/ADBE	2.10	120481.29	8	15060.16	0.24	15654.84	59	265.34
	EBAY/ADBE	0.09	4298.03	9	477.56	-3.34	-183978.63	50	-3679.57
MT	IGE/SPY/ADBE	1.24	48597.14	7	6942.45	-1.13	-51317.98	27	-1900.67

PT - Pairs Trading, MT - Multiple Trading
TRAN - Number of transactions, PP Sig - Profit per signal,
ASR - Annualized Sharpe ratio, DDIVF - Data-driven innovation volatility forecast

3.6.8 Summary of the three approaches

Table 3.5 summarizes the **highest profits** attained under the three approaches (cointegration, PageRank with P_t and PageRank with d_t) for the 6 windows. In order to compare the profits, the table looks at per-transaction profits rather than cumulative profits because it considers the number of transactions that occurred over a given period of time. The two proposed methods with the PageRank algorithm have been able to identify more profitable trading strategies than the cointegration approach. During the pre-COVID period, the highest profit of 10574.31 is attained by ADBE and SPY with the PageRank on P_t series. During the COVID period, the highest profit of 20270.49 is attained by pairs trading on GDX and ADBE with the PageRank on d_t series. During the post-COVID period, the highest profit of 31491.49 is attained by pairs trading on IGE and ADBE with the PageRank on the d_t series.

Table 3.5: Summary table of profits made by DDIVF method

Period	Time window	Cointegration	PageRank with P_t	PageRank with d_t
Pre-COVID	2	4073.64	2234.38	4073.60
	4	5403.03	10574.31	9480.61
COVID	8	18762.15	14175.55	20270.49
	10	1226.27	11630.57	5104.51
Post-COVID	14	10534.75	17884.01	31491.49
	16	3783.62	4088.95	15060.16

3.7 Conclusion

Chapter 3 discusses a novel method for asset selection based on network graphs and their PageRank scores for pairs and multiple trading. The existing approach, asset selection based on cointegration, is compared with our novel approach in terms of ratio between cumulative profit and number of transactions occurred during the considered time windows. Moreover, we present two methods of estimating innovation volatility, which is necessary in generating trading signals in profits determination.

This study addresses a gap in applications that compare the performance of pairs trading based on different stock selection techniques. We have adopted dynamic financial networks based on the correlation between price series (P_t) and financial networks based on the correlation between first differenced d_t series to rank stocks (nodes). Stocks with adjacent PageRank scores are used to form different trading strategies to apply in pairs and multiple trading.

To illustrate the practical application of the proposed methods, we used real price series data extracted from Yahoo! Finance during three different time periods, pre-COVID, COVID and post-COVID, considering their varying trend and volatility observed. Trading strategies formed using all three approaches (cointegration, PageRank with P_t and PageRank with d_t) provide higher per-transaction profits with the DDIVF method compared to the $\sqrt{Q_t}$ method of estimating innovation volatility. In each window, the most profitable strategy is formed by either the second (P_t) or the third (d_t) approach. Hence, in these

cases, financial network ranking methods provide the opportunity to identify more profitable combinations of the stocks than the cointegration approach. It is worth noting that these results are data-driven and the profits are realistic only when we apply trading algorithms to assets which are cointegrated or correlated in a previous time window. This being said,we note that this example is just for educational purposes. We would advise against implementing this strategy in live trading without a comprehensive understanding of its mechanics and risks.

Finally, in this study, our main focus was to identify a novel method to select assets for pairs and multiple trading and compare its profitability with that of the assets selected under the cointegration approach. There are opportunities for future research to extend the profits determination by computing the market transaction cost.

Contributions

Listed below are the conference publications made under my contribution.

- **T. Ranathungage, S. Bowala, E. Hoque, A. Thavaneswaran and R.K. Thulasiram, "Application of a Novel Fuzzy Pattern Mining Algorithm for Sequence Data", Proceedings of 48th IEEE International Conference on Computers, Software, and Applications (COMPSAC 2024).**

- **T. Ranathungage, A. Thavaneswaran, Y. Liang, R.K. Thulasiram and A. Paseka, "Novel Data-Driven Dynamic Network Science Application in Algorithmic Trading", Proceedings of 48th IEEE International Conference on Computers, Software, and Applications (COMPSAC 2024).**

- **Y. Liang, A. Thavaneswaran, J Liyau, A. Muhamad, T. Ranathungage and R.K. Thulasiram, "A Cryptocurrency Multiple Trading Strategy with Kalman Filter Innovation Volatility Interval Forecasts", Proceedings of 48th IEEE International Conference on Computers, Software, and Applications (COMPSAC 2024).**

- **S. Bowala, A. Thavaneswaran, R. Thulasiram, T. Ranathungage and J. D. Das, "High Frequency Data-Driven Dynamic Portfolio Optimization for Cryptocurrencies," 2023 IEEE Symposium Series on Computational Intelligence (SSCI), Mexico City, Mexico, 2023, pp. 375-380, doi: 10.1109/SSCI52147.2023.10371936.**

Appendix A

A.1 Fuzzy Set Theory

This section provides an introduction to fuzzy set theory, as discussed in Thavaneswaran et al. (2009). Many researchers are particularly interested in using fuzzy set theory as a methodology for modelling and analysing financial problems, which involve imprecision and vagueness because of its capability to quantitatively and qualitatively handle those problems. Trapezoidal fuzzy numbers (Tr.F.N.) and triangular fuzzy numbers (T.F.N.) are two examples of linear fuzzy numbers, and they are typically used to model parameters in the majority of fuzzy financial models that have been developed thus far. The primary motivation behind utilising a linear membership function is to get rid of intricate nonlinear calculations.

A.1.1 Preliminaries and notations

Definition A.1.1.a. A fuzzy set A in $X \subset \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers, is a set of ordered pairs $A = \{(x; \mu(x)) : x \in X\}$, where $\mu(X)$ is the

membership function, the degree of compatibility or degree of truth of $x \in X$, which maps to the real interval $[0, 1]$.

Definition A.1.1.b. A fuzzy number $\tilde{A} \in \mathbb{F}$ is called a trapezoidal fuzzy number (Tr.F.N.) with core $[a, b]$, left width α and right width β if its membership function has the following form:

$$\mu(t) = \begin{cases} 1 - \frac{\alpha - t}{\alpha} & \text{if} \quad a - \alpha \leq t \leq \alpha, \\ 1 & \text{if} \quad a \leq t \leq b, \\ 1 - \frac{t - b}{\beta} & \text{if} \quad a \leq t \leq b + \beta, \\ 0 & \text{otherwise.} \end{cases}$$

We use the notation $\tilde{A} = (a, b, \alpha, \beta)$. It can be shown that,

$$A(\gamma) = [a_1(\gamma), a_2(\gamma)] = [a - (1 - \gamma)\alpha, b + (1 - \gamma)\beta] \quad \forall \gamma \in [0, 1]. \tag{A.1}$$

The support of $\tilde{A}$ is $(a - \alpha, b - \beta)$. Moreover, for any fuzzy number $\tilde{A}$ and a positive real number C, where the following relationship holds

$\tilde{A} \leq C \iff \int_0^1 (a_1(\gamma) + a_2(\gamma))\gamma \, d\gamma \leq C.$

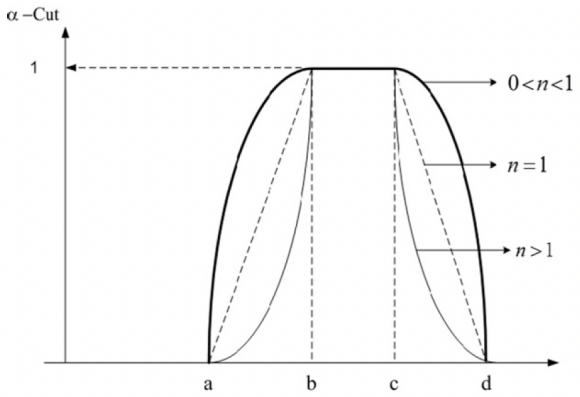


Figure A.1: Nonlinear membership function

Source: Thavaneswaran et al. (2009)

Definition A.1.1.c. A fuzzy number $A(x), x \in \mathbb{R}$ is of the form

$$A(x) = \begin{cases} g(x) & \text{when} \quad x \in [a, b), \\ 1 & \text{when} \quad x \in [b, c), \\ h(x) & \text{when} \quad x \in [c, d), \\ 0 & \text{otherwise}, \end{cases} \tag{A.2}$$

where g is a real, increasing and right continuous function, h is a real, decreasing and left continuous function, and $a; b; c; d$ are real numbers such that $a < b < c < d$. Notice that the Equation A.2 is an LR fuzzy number with strictly monotone shape functions as described in Bodjanova (2005). A fuzzy number A with shape functions g and h defined by

$$g(x) = \left(\frac{x - a}{b - a} \right)^m \tag{A.3}$$

and

$$h(x) = \left(\frac{d-x}{d-c}\right)^n \tag{A.4}$$

respectively, where m or $n > 0$, is denoted by $A = [a, b, c, d]_{m,n}$. If m and $n = 1$, we simply write $A = [a, b, c, d]$, which is known as a trapezoidal fuzzy number (Linear type fuzzy number). If m or $n \neq 1$, a fuzzy number $A^* = [a, b, c, d]_{m,n}$ is a modification of a trapezoidal fuzzy number (nonlinear type fuzzy number) $A = [a, b, c, d]_{m,n}$. If m and $n > 1$, then A^* is a concentration of A as in Figure A.1. Concentration of A by m and $n = 2$ is often interpreted as the linguistic hedge "very". If $0 < (m\, or\, n) < 1$, then A^* is a dilation of A. Dilation of A by m and $n = 0.5$ is often interpreted as the linguistic hedge "more or less". Each fuzzy number A described by Equation (A.3) and Equation (A.4) has the following γ-level sets, $A(\gamma) = [a(\gamma), b(\gamma)], a(\gamma), b(\gamma) \in \mathbb{R}, \gamma \in [0, 1]$ and

$$A(\gamma) = [g^{-1}(\gamma), h^{-1}(\gamma)], \quad A_1 = [b, c], \quad A_0 = [a, d].$$

If $A = [a, b, c, d]_{m,n}$ then, $\forall\, \alpha \in [0, 1]$,

$$A(\gamma) = [a + \gamma^{\frac{1}{m}}(b-a), d - \gamma^{\frac{1}{n}}(d-c)]. \tag{A.5}$$

Weighted possibilistic moments (WPM)

The first order f -WPM(or weighted possibilistic mean) of $A \in \mathbb{F}$ is given by

$$\overline{M}_f(A) = \int_0^1 f(\gamma)\frac{(a_1(\gamma) + a_2(\gamma))}{2}d\gamma, \tag{A.6}$$

where $f(\gamma)$ is a weight function such that $\int_0^1 f(\gamma)d\gamma = 1$. Similarly, the centered WPM (or weighted variance) of $A \in \mathbb{F}$ is

$$Var_f(A) = \frac{1}{2}\int_0^1 f(\gamma)[(a_1(\gamma) - \overline{M}_f(A))^2 + (a_2(\gamma) - \overline{M}_f(A))^2]d\gamma, \tag{A.7}$$

and for any positive integer r, the f -WPM of order r about the possibilistic mean value of A is defined as,

$$E_r(A) = \frac{1}{2}\int_0^1 f(\gamma)[(a_1(\gamma) - \overline{M}_f(A))^r + (a_2(\gamma) - \overline{M}_f(A))^r]d\gamma. \tag{A.8}$$

In analogy with Thavaneswaran et al. (2007), the f -weighted possibilistic skewness and the f-weighted possibilistic kurtosis of a fuzzy number A are defined as

$$Skewness_f(A) = \frac{E(A - \overline{M}_f(A))}{(\sqrt{Var_f(A)})^3} \tag{A.9}$$

$$K^{(A)} = \frac{E_4(A)}{(E_2(A))^2}. \tag{A.10}$$

Lemma A.1 Let the γ-level sets of a Tr.F.N be given by Equation A.1,

Then the possibilistic moments are given by:

I. $\overline{M}(A) = \frac{\beta-\alpha}{6} + \frac{a+b}{2}$

Proof:

$$\begin{aligned}
\overline{M}(A) &= \int_0^1 \gamma(a-(1-\gamma)\alpha + b + (1-\gamma)\beta)d\gamma \\
&= a\int_0^1 \gamma d\gamma - \alpha\int_0^1 (1-\gamma)\gamma d\gamma + b\int_0^1 \gamma d\gamma + \beta\int_0^1 (1-\gamma)\gamma d\gamma \\
&= \frac{a}{2} - \frac{\alpha}{2} + \frac{\alpha}{3} + \frac{b}{2} + \frac{\beta}{2} - \frac{\beta}{3} \\
&= \frac{a+b}{2} + \frac{\beta-\alpha}{6}.
\end{aligned}$$

II. $Var(A) = \frac{\alpha\beta+\alpha^2+\beta^2}{18} + \frac{(\alpha+\beta)(b-a)}{6} + \frac{a^2+b^2}{4} - \frac{ab}{2}$

Proof:

$$\begin{aligned}
Var(A) &= \int_0^1 \gamma\left[\left(a-(1-\gamma)\alpha - \left(\frac{a+b}{2} + \frac{\beta-\alpha}{6}\right)\right)^2 + \left(b+(1-\gamma)\beta - \left(\frac{a+b}{2} + \frac{\beta-\alpha}{6}\right)\right)^2\right]d\gamma \\
&= \int_0^1 \gamma\left[\frac{a-b}{2} - (1-\gamma)\alpha - \left(\frac{\beta-\alpha}{6}\right)\right]^2 d\gamma + \int_0^1 \gamma\left[\frac{b-a}{2} + (1-\gamma)\beta - \left(\frac{\beta-\alpha}{6}\right)\right]^2 d\gamma
\end{aligned}$$

The first integral in the last expression $\left(\int_0^1 \gamma\left[\frac{a-b}{2} - (1-\gamma)\alpha - \left(\frac{\beta-\alpha}{6}\right)\right]^2 d\gamma\right)$ can be calculated as,

$$= \left(\frac{a-b}{2}\right)^2 \int_0^1 \gamma d\gamma + \alpha^2 \int_0^1 \gamma(1-\gamma)^2 d\gamma + \left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma d\gamma$$

$$- 2 * \left(\frac{a-b}{2}\right) \alpha \int_0^1 \gamma(1-\gamma) d\gamma - 2 * \left(\frac{a-b}{2}\right) \left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma d\gamma$$

$$+ 2 * \left(\frac{\beta-\alpha}{6}\right) \alpha \int_0^1 \gamma(1-\gamma) d\gamma.$$

$$= \frac{1}{2}\left(\frac{a-b}{2}\right)^2 + \alpha^2 \left(\frac{1}{12}\right) + \frac{1}{2}\left(\frac{\beta-\alpha}{6}\right)^2$$

$$- (a-b)\alpha \left(\frac{1}{6}\right) - \left(\frac{a-b}{2}\right)\left(\frac{\beta-\alpha}{6}\right) + \left(\frac{\beta-\alpha}{3}\right)\alpha \left(\frac{1}{6}\right).$$

Also, the second integral of $Var(A)$, $\int_0^1 \gamma \left[\frac{b-a}{2} + (1-\gamma)\beta - \left(\frac{\beta-\alpha}{6}\right)\right]^2 d\gamma$ can be calculated as,

$$= \left(\frac{b-a}{2}\right)^2 \int_0^1 \gamma d\gamma + \beta^2 \int_0^1 \gamma(1-\gamma)^2 d\gamma + \left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma d\gamma$$

$$+ 2 * \left(\frac{b-a}{2}\right) \beta \int_0^1 \gamma(1-\gamma) d\gamma - 2 * \left(\frac{b-a}{2}\right) \left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma d\gamma$$

$$- 2 * \left(\frac{\beta-\alpha}{6}\right) \beta \int_0^1 \gamma(1-\gamma) d\gamma.$$

$$= \frac{1}{2}\left(\frac{b-a}{2}\right)^2 + \beta^2\left(\frac{1}{12}\right) + \frac{1}{2}\left(\frac{\beta-\alpha}{6}\right)^2 + (b-a)\beta\left(\frac{1}{6}\right)$$

$$- \left(\frac{b-a}{2}\right)\left(\frac{\beta-\alpha}{6}\right) - \left(\frac{\beta-\alpha}{3}\right)\beta\left(\frac{1}{6}\right).$$

Therefore,

$$Var(A) = \left(\frac{a-b}{2}\right)^2 + \frac{\alpha^2+\beta^2}{12} + \left(\frac{\beta-\alpha}{6}\right)^2 - (a-b)\left(\frac{\alpha}{6}+\frac{\beta}{6}\right) + \left(\frac{\beta-\alpha}{3}\right)\left(\frac{\alpha}{6}-\frac{\beta}{6}\right)$$

$$= \frac{\alpha\beta+\alpha^2+\beta^2}{18} + \frac{(\alpha+\beta)(b-a)}{6} + \frac{a^2+b^2}{4} - \frac{ab}{2}.$$

III. $Skewness(A) = \frac{19(\beta^3-\alpha^3)}{1080} + \frac{(\alpha^2-\beta^2)(a-b)}{24} + \frac{\alpha\beta(\beta-\alpha)}{72}$.

Proof:

$$Skew(A) = \int_0^1 \gamma\left[(a_1(\gamma) - M(A))^3 + (a_2(\gamma) - M(A))^3\right] d\gamma$$

$$= \int_0^1 \gamma\left[\left(a - (1-\gamma)\alpha - \left(\frac{a+b}{2} + \frac{\beta-\alpha}{6}\right)\right)^3 + \left(b + (1-\gamma)\beta - \left(\frac{a+b}{2} + \frac{\beta-\alpha}{6}\right)\right)^3\right] d\gamma$$

$$= \int_0^1 \gamma\left[\frac{a-b}{2} - (1-\gamma)\alpha - \left(\frac{\beta-\alpha}{6}\right)\right]^3 d\gamma + \int_0^1 \gamma\left[\frac{b-a}{2} + (1-\gamma)\beta - \left(\frac{\beta-\alpha}{6}\right)\right]^3 d\gamma$$

The first integral in the last expression, $\int_0^1 \gamma\left[\frac{a-b}{2} - (1-\gamma)\alpha - \left(\frac{\beta-\alpha}{6}\right)\right]^3 d\gamma$,

can be calculated as,

$$= \left(\frac{a-b}{2}\right)^3 \int_0^1 \gamma d\gamma - \alpha^3 \int_0^1 \gamma(1-\gamma)^3 d\gamma - \left(\frac{\beta-\alpha}{6}\right)^3 \int_0^1 \gamma d\gamma$$

$$- 3\alpha \left(\frac{a-b}{2}\right)^2 \int_0^1 \gamma(1-\gamma) d\gamma - 3 \left(\frac{a-b}{2}\right)^2 \left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma d\gamma$$

$$+ 3 \left(\frac{a-b}{2}\right) \alpha^2 \int_0^1 \gamma(1-\gamma)^2 d\gamma - 3\alpha \left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma(1-\gamma) d\gamma$$

$$+ 3 \left(\frac{a-b}{2}\right) \left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma d\gamma - 3\alpha^2 \left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma(1-\gamma)^2 d\gamma$$

$$+ 6\alpha \left(\frac{a-b}{2}\right) \left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma(1-\gamma) d\gamma.$$

$$= \frac{1}{2}\left(\frac{a-b}{2}\right)^3 - \alpha^3 \left(\frac{1}{20}\right) - \frac{1}{2}\left(\frac{\beta-\alpha}{6}\right)^3 - \frac{1}{2}\alpha \left(\frac{a-b}{2}\right)^2 - \frac{3}{2}\left(\frac{a-b}{2}\right)^2 \left(\frac{\beta-\alpha}{6}\right)$$

$$+ \frac{1}{4}\alpha^2 \left(\frac{a-b}{2}\right) - \frac{1}{2}\alpha \left(\frac{\beta-\alpha}{6}\right)^2 + \frac{3}{2}\left(\frac{a-b}{2}\right)\left(\frac{\beta-\alpha}{6}\right)^2 - \frac{1}{4}\alpha^2 \left(\frac{\beta-\alpha}{6}\right)$$

$$+ \alpha \left(\frac{a-b}{2}\right)\left(\frac{\beta-\alpha}{6}\right).$$

Also, the second integral of $Skew(A)$, $\int_0^1 \gamma \left[\frac{b-a}{2} + (1-\gamma)\beta - \left(\frac{\beta-\alpha}{6}\right)\right]^3 d\gamma$, can be calculated as,

$$= \left(\frac{b-a}{2}\right)^3 \int_0^1 \gamma d\gamma + \beta^3 \int_0^1 \gamma(1-\gamma)^3 d\gamma - \left(\frac{\beta-\alpha}{6}\right)^3 \int_0^1 \gamma d\gamma$$

$$+ 3\left(\frac{b-a}{2}\right)\beta^2 \int_0^1 \gamma(1-\gamma)^2 d\gamma + 3\beta\left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma(1-\gamma) d\gamma$$

$$- 3\left(\frac{\beta-\alpha}{6}\right)\left(\frac{b-a}{2}\right)^2 \int_0^1 \gamma d\gamma + 3\left(\frac{b-a}{2}\right)^2 \beta \int_0^1 \gamma(1-\gamma) d\gamma$$

$$- 3\beta^2\left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma(1-\gamma)^2 d\gamma + 3\left(\frac{b-a}{2}\right)\left(\frac{\beta-\alpha}{6}\right)^2 \int_0^1 \gamma d\gamma$$

$$- 6\beta\left(\frac{b-a}{2}\right)\left(\frac{\beta-\alpha}{6}\right) \int_0^1 \gamma(1-\gamma) d\gamma.$$

$$= -\frac{1}{2}\left(\frac{a-b}{2}\right)^3 + \beta^3\left(\frac{1}{20}\right) - \frac{1}{2}\left(\frac{\beta-\alpha}{6}\right)^3 - \frac{1}{4}\beta^2\left(\frac{a-b}{2}\right) + \frac{1}{2}\beta\left(\frac{\beta-\alpha}{6}\right)^2$$

$$- \frac{3}{2}\left(\frac{a-b}{2}\right)^2\left(\frac{\beta-\alpha}{6}\right) + \frac{1}{2}\beta\left(\frac{a-b}{2}\right)^2 - \frac{1}{4}\beta^2\left(\frac{\beta-\alpha}{6}\right) - \frac{3}{2}\left(\frac{a-b}{2}\right)\left(\frac{\beta-\alpha}{6}\right)^2$$

$$+ \beta\left(\frac{a-b}{2}\right)\left(\frac{\beta-\alpha}{6}\right).$$

Therefore,

$$
\begin{aligned}
Skew(A) = \frac{\beta^3 - \alpha^3}{20} - \left(\frac{\beta - \alpha}{6}\right)^3 - \frac{1}{4}\left(\beta^2 - \alpha^2\right)\left(\frac{a-b}{2}\right) \\
+ \frac{1}{2}\left(\beta - \alpha\right)\left(\frac{\beta - \alpha}{6}\right)^2 - 3\left(\frac{a-b}{2}\right)^2\left(\frac{\beta - \alpha}{6}\right) + \frac{1}{2}\left(\beta - \alpha\right)\left(\frac{a-b}{2}\right)^2 \\
- \frac{1}{4}\left(\beta^2 + \alpha^2\right)\left(\frac{\beta - \alpha}{6}\right) + \left(\beta + \alpha\right)\left(\frac{a-b}{2}\right)\left(\frac{\beta - \alpha}{6}\right) \\
= \frac{19(\beta^3 - \alpha^3)}{1080} + \frac{(\alpha^2 - \beta^2)(a-b)}{24} + \frac{\alpha\beta(\beta - \alpha)}{72}.
\end{aligned}
$$

Appendix B

B.1 Chapter 2 codes

```
## Successive Occurrences of DNA Nucleotides
## dataset bnrf1ebv (Nucleotide sequence - BNRF1 Epstein-Barr)
library(markovchain)
library(astsa)
library(seqinr)

## bnrf1ebv, bnrf1hvs - with numbers  (***and EBV - with
    letters)
data = bnrf1ebv
for(i in 1:length(data)){
  if(data[i]==1){
    data[i] = "A"
  }
  else if(data[i]==2){
    data[i] = "C"
  }
  else if(data[i]==3){
    data[i] = "G"
  }
  else{
    data[i] = "T"
  }
}


## Base Proportions of Nucleotide in bnrf1ebv
alp <- c("A","C","G","T")
zeroOrderFreq <- count(data,1,alphabet=alp,freq=TRUE)
```

```
barplot(zeroOrderFreq,col=1:4,
        xlab="Nucleotide",
        ylab="Base proportion")

## Base Proportions of dinucleotide in bnrf1ebv
count(data,2,alphabet=alp,freq=FALSE)
zeroOrderFreqDin <- count(data,2,alphabet=alp,freq=TRUE)
round(zeroOrderFreqDin,3)
barplot(zeroOrderFreqDin,col=rainbow(16),
        xlab="Dinucleotide",
        ylab="Base proportion", las = 2)

## transition probability matrix
PtildaTemp = markovchainFit(data)
Ptilda = markovchainFit(data)$estimate
Ptilda = as.matrix(Ptilda[1:4])
round(Ptilda,3)

## Check lower end and upper end matrices
Ptilda_lower = markovchainFit(data)$lowerEndpointMatrix
round(Ptilda_lower,3)
Ptilda_upper = markovchainFit(data)$upperEndpointMatrix
round(Ptilda_upper,3)

## Matrix Approach
## pattern matrix for bnrf1ebv as accgc the pattern
t<-c(0.825,0.175,0,0,0,0)
a<-c(0.531,0.175,0.294,0,0,0)
ac<-c(0.445,0.237,0,0.318,0,0)
acc<-c(0.544,0.237,0,0,0.219,0)
accg<-c(0.504,0.198,0,0,0,0.298)
accgc<-c(0,0,0,0,0,1)
P<-matrix(c(t,a,ac,acc,accg,accgc),nr=6,nc=6,byrow = T)
P
Q<-P[1:5,1:5]
F= solve(diag(5)-Q)  ## F = (I-Q)^(-1)
F

## average number of nucleotides needed to reach pattern accgc
   in bnrf1hvs
round(sum(F[1,]),3)



## Function to Calculate Expected Waiting Time
```

```
ExpWaitTime <- function(labels = c(0,1), trials = 100, pattern = c(0,0,1,1,1),
              P = matrix(c(0.25, 0.25, 0.25, 0.25), nrow = 2, ncol = 2)){
  init <- as.vector(rep(1/length(labels),length(labels)))
  states <- 1:length(init)
  simlist <- c()
  simlist1 <- c()
   for (i in 1:trials) {
   simlist[1] <- sample(states,1,prob=init)
    for (j in 2:length(pattern)) {
      simlist[j] <- sample(states,1,prob=P[simlist[j-1],])
    }
    STATE <- c()
    for (k in 1:length(pattern)) {
      STATE[k] = simlist[k]
    }
    n<-length(pattern)
    m<-length(pattern)
    temp=n-2
    while (!prod(labels[STATE]==pattern)){
      simlist[n+1] <- sample(states,1,prob=P[simlist[n],])
      for (l in 1:length(pattern)) {
        STATE[l] = simlist[n-temp+l-1]
      }
      n = n+1
      m = m+1
    }
    simlist1[i] <- m
    }
  return(mean(simlist1))
  }

## Function to find the the power (k^th power) of a given matrix
MatrixPower <- function(Matrix,k) {
  if (k == 0) return (diag(dim(Matrix)[1]))
  if (k == 1) return(Matrix)
  if (k > 1) return(Matrix %*% MatrixPower(Matrix, k-1))
}

## Finding Stationary probabiities for a given matrix

## Stationary distribution of discrete-time Markov chain
## (uses eigenvectors)
```

```
stationary <- function(mat){
  x = eigen(t(mat))$vectors[,1]
  return(as.double(x/sum(x)))
}
## Fuzzy pattern mining algorithm
FuzzyExpWaitTime <- function(labels = c(0,1), Ntrials = 100,
        pattern = c(0,0,1,1,1), Ptilda = matrix(c(0.25 ,0.25,
    0.25, 0.25),
        nrow = 2, ncol = 2), alpha = seq(0.1,0.9,0.1), MMValue
    = 3608.52){

  ## Initializing minimum alpha
  alpha_min <- 0
  ## Initializing stochastic matrix (each entry is same and
    represent
  #equal probability)
  A <- matrix(1/length(labels), nrow = length(labels), ncol =
    length(labels))
  ## Initializing temporary vector to save expected wait time
    for each alpha
  TempExpTime <- c()
  ## Maximum matrix power (l). For now we set maximum matrix
    power to
  #length of the pattern we are untested in.
  l = length(pattern)
  ## Note: For each i (i.e. alpha value), we have 5 different
  #powers (DNA example) of TempMatrix. Thus, to save all the
    matrices
  #in correct order (alpha value and power), we need to save
    matrices within
  #a matrix.

  ## Initializing a matrix which has a matrix in each entry
  m <- matrix(0, nrow = length(labels), ncol = length(labels))
  M <- matrix(list(m), nrow = length(alpha), ncol = l)
  ## For loop to calculate expected wait time and matrix power
    for each alpha
  for (i in 1:length(alpha)) {
  ## Fuzzyfying
  TempMatrix = alpha[i]*A + (1-alpha[i])*Ptilda

## Calculate expected wait time using "ExpWaitTime" for each
    alpha
TempExpTime[i] <- ExpWaitTime(labels = labels, trials = Ntrials
    ,
```

```
148                                     pattern = pattern, P =
    TempMatrix)
149     ## Finding matrix powers
150     for (j in 1:l) {
151       M[[i,j]] <- MatrixPower(TempMatrix,j)
152     }
153     }
154
155   ## Finding alpha that gives the expected waiting time closer
     to
156   #the book value
157   alpha_min <- alpha[which.min(abs(TempExpTime-MMValue))]
158   ExpWaitTime <- TempExpTime[which.min(abs(TempExpTime-MMValue)
     )]
159
160   ## Return Values:
161   ## 1. alpha that gives the expected waiting time closer to
     the book value
162   ## 2. expected waiting time closer to the book value
163   ## 3. matrix powers of TempMatrix for each alpha
164   return(list(alpha_min=alpha_min, ExpWaitTime=ExpWaitTime,
     MatrixPower=M))
165 }
166
167 ## Fuzzy pattern mining algorithm for bnrf1ebv data
168 FuzzyResultsDNA = FuzzyExpWaitTime(labels = c(1,2,3,4), Ntrials
     = 1000,
169                   pattern = c(1,2,2,3,2), P = Ptilda,
170                   alpha <- seq(0.2,0.3,0.001),
171                   MMValue = 913.601) ##mm value is matrix
    approach value
172
173 ## Print the alpha value that gives the expected waiting time
    closer
174 #to the book value
175 FuzzyResultsDNA$alpha_min
176 ## Expected waiting time closer to the book value
177 FuzzyResultsDNA$ExpWaitTime
178 ## Print the matrix power of 5 for 1st alpha value (i.e. alpha
    = 0.1)
179 FuzzyResultsDNA$MatrixPower[1,5]
180 ## Print the matrix power of 4 for 3rd alpha value (i.e. alpha
    = 0.3)
181 FuzzyResultsDNA$MatrixPower[3,4]
182
```

```
183 ## Observe and Estimated proportions and counts
184 ## Observe counts
185 count(data,1,alphabet=alp,freq=FALSE)
186
187 ## Observe Proportions
188 zeroOrderFreq <- count(data,1,alphabet=alp,freq=TRUE)
189 round(zeroOrderFreq,3)
190
191 ## Estimated proportions
192 LongTermProb_DNA = stationary(Ptilda)
193 round(LongTermProb_DNA,3)
194
195 ## Estimated counts
196 round(LongTermProb_DNA*length(data),3)
197
198 ##Application 2
199 ## In computer security applications, a honeypot is a trap set
    on
200 #a network to detect and counteract computer hackers.
201 #Honeypot data are studied in Kimou et al. (2010) using Markov
    chains.
202 #The authors obtain honeypot data from a central database
203 #and observe attacks against four computer ports 80, 135,
    139,
204 #and 445 over 1 year. The ports are the states of a Markov
    chain
205 #along with a state corresponding to no port is attacked.
    Weekly data are
206 #monitored, and the port most often attacked during the week is
     recorded.
207 #The estimated Markov transition matrix for weekly attacks is
208
209
210 Ptilda <- matrix(0, nrow = 5, ncol = 5)
211 colnames(Ptilda) <- rownames(Ptilda) <- c(1,2,3,4,5) # 1=80,
    2=135, 3=139, 4=445, 5=No attack
212 Ptilda[1,] <- c(0, 0, 0, 0, 1)  # Given an 80 in the 1st
    position
213 Ptilda[2,] <- c(0, 8/13, 3/13, 1/13, 1/13)  # Given a 135 in
    the 1st position
214 Ptilda[3,] <- c(1/16, 3/16, 3/8, 1/4, 1/8) # Given a 139 in the
     1st position
215 Ptilda[4,] <- c(0, 1/11, 4/11, 5/11, 1/11)  # Given a 445 in
    the 1st position
216 Ptilda[5,] <- c(0, 1/8, 1/2, 1/8, 1/4)  # Given a No attack in
```

```
   the 1st position
Ptilda
round(MatrixPower(Ptilda,2),3)
## Finding Expected Waiting Time for no attacks for 4
   consecutive weeks
WaitTimeSecurity <- ExpWaitTime(labels = c(1,2,3,4,5), trials =
    100,
                                pattern = c(5,5,5,5), P =
   Ptilda)
WaitTimeSecurity

## Matrix appraoch
## An absorbing Markov chain is constructed with transition
   matrix
## Pattern interested: (5555)
## "A" correspond to null event

A = c(3/4,1/4,0,0,0)
NA1 = c(3/4,0,1/4,0,0)
NA2 = c(3/4,0,0,1/4,0)
NA3 = c(3/4,0,0,0,1/4)
NA4 = c(0,0,0,0,1)
P <- matrix(c(A,NA1,NA2,NA3,NA4),nr=5,nc=5,byrow = T)
P
Q<-P[1:4,1:4]
F= solve(diag(4)-Q)  ## F = (I-Q)^(-1)
F

## Expected Waiting Time (Matrix Approach) for 5555. i.e. no
   attacks for a
#month
MatrixMethod_ComSecirity = sum(F(1,))
MatrixMethod_ComSecirity
## Fuzzyfying
FuzzySecurity <- FuzzyExpWaitTime(labels = c(1,2,3,4,5),
   Ntrials = 20,
                                  pattern = c(5,5,5,5), Ptilda
   = Ptilda,
                                  alpha <- seq(0.2,0.4,0.001),
   MMValue = 340)
FuzzySecurity$alpha_min
FuzzySecurity$ExpWaitTime
FuzzySecurity$MatrixPower[192,4] # conditional probabilities
   after 4 weeks
#for alpha = 0.391
```

```
## Stationary probabilities
LongTermProb_ComSeq = stationary(Ptilda)
LongTermProb_ComSeq

## Application 3
start.date = '2017-01-01'
end.date = '2022-11-30'
getSymbols(c("^GSPC","^VIX","GOOG","AAPL"), src = "yahoo",
           from = start.date, to = end.date)

stocks <- merge(SP500 = GSPC[, "GSPC.Close"],VIX = VIX[, "VIX.Close"],
                GOOG = GOOG[, "GOOG.Close"],AAPL = AAPL[, "AAPL.Close"])
head(stocks)

assets <- c('SP500','VIX','GOOG','AAPL')
xy <- stocks
colnames(xy) <- assets
##Calculate log returns

log_return<-xy %>% log %>% diff #log returns
head(log_return)
tail(log_return)
nrow(log_return)

##Creating signs
signal <- list()
for (i in 1:length(assets)){
  signal [[i]] <- ifelse(as.data.frame(log_return[, i]) >= 0, 1, -1)#if the
  #ith col of logreturns >=0 (positive) then create signal with 1,
  #otherwise (logreturns<0) signal=-1
}
signal<-as.data.frame(do.call(cbind, signal))
colnames(signal) <- assets
head(signal)

##Creating markov matrix
markovmatrix <-list()
for (j in 1:length(assets)){
  seq1 = na.omit (as.vector (signal[, j]))
  markovmatrix[[j]] <-markovchainFit(seq1)$estimate
}
```

```
asset_number<-3
assets[asset_number]
stock<-as.matrix(markovmatrix[[asset_number]][1:2])# goog.
#Change first argument to get matrix for each CC.
#i.e.markovmatrix[[2]][1:2] for vix.
colnames(stock) <- c(0,1) #Assign names to the columns
rownames(stock) <- c(0,1)
stock
##Matrix approach- pattern 0101-aapl
phi<-c(0.552,0.448,0,0,0)
s1<-c(0,0.448,0.552,0,0)
s2<-c(0.521,0,0,0.479,0)
s3<-c(0,0.448,0,0,0.552)
s4<-c(0,0,0,0,1)
P<-matrix(c(phi,s1,s2,s3,s4),nr=5,nc=5,byrow = T)
P
Q<-P[1:4,1:4]
F= solve(diag(4)-Q)  ## F = (I-Q)^(-1)
F
## New Approach:Fuzzy Pattern Mining Algorithm
## Function to Calculate Expected Waiting Time
ExpWaitTime <- function(labels = c(0,1), trials = 1000, pattern
    = c(0,0,1,1),
                        P = matrix(c(0.25, 0.25, 0.25, 0.25),
                                   nrow = 2, ncol = 2)){
  init <- as.vector(rep(1/length(labels),length(labels)))
  states <- 1:length(init)
  simlist <- c()
  simlist1 <- c()
  for (i in 1:trials) {
    simlist[1] <- sample(states,1,prob=init)
    for (j in 2:length(pattern)) {
      simlist[j] <- sample(states,1,prob=P[simlist[j-1],])
    }
    STATE <- c()
    for (k in 1:length(pattern)) {
      STATE[k] = simlist[k]
    }
    n<-length(pattern)
    m<-length(pattern)
    temp=n-2
    while (!prod(labels[STATE]==pattern)){
      simlist[n+1] <- sample(states,1,prob=P[simlist[n],])
      for (l in 1:length(pattern)) {
        STATE[l] = simlist[n-temp+l-1]
```

```
337       }
338       n = n+1
339       m = m+1
340     }
341     simlist1[i] <- m
342     }
343   return(mean(simlist1))
344   }
345
346 ## Fuzzyfying: (P = pQ + (1     p)A)
347 ## Function to find the the power (k^th power) of a given
      matrix
348 MatrixPower <- function(Matrix,k) {
349   if (k == 0) return (diag(dim(Matrix)[1]))
350   if (k == 1) return(Matrix)
351   if (k > 1) return(Matrix %*% MatrixPower(Matrix, k-1))
352 }
353
354 FuzzyExpWaitTime <- function(labels = c(0,1), trials = 1000,
355                      pattern = c(0,0,1,1), P = matrix(c(0.25
      ,0.25, 0.25, 0.25),
356                      nrow = 2, ncol = 2),alpha = seq
      (0.1,0.9,0.1),
357                      MMValue =26.43){
358   ## Initializing minimum alpha
359   alpha_min <- 0
360   ## Initializing stochastic matrix (each entry is same and
361   #represent equal probability)
362   R <- matrix(1/length(labels), nrow = length(labels), ncol =
       length(labels))
363   ## Initializing temporary vector to save expected wait time
       for each alpha
364   TempExpTime <- c()
365   ## Maximum matrix power (k). For now we set maximum matrix
       power to
366   #length of the pattern we are interested in.
367   k = length(pattern)
368   ## Initializing a matrix which has a matrix in each entry
369   m <- matrix(0, nrow = length(labels), ncol = length(labels))
370   M <- matrix(list(m), nrow = length(alpha), ncol = k)
371   ## For loop to calculate expected wait time and matrix power
       for each alpha
372   for (i in 1:length(alpha)) {
373     ## Fuzzyfying
374     TempMatrix = alpha[i]*R + (1-alpha[i])*P
```

```
375     ## Calculate expected wait time using "ExpWaitTime" for
    each alpha
376     TempExpTime[i] <- ExpWaitTime(labels = labels, trials =
    trials,
377                                   pattern = pattern, P =
    TempMatrix)
378     ## Finding matrix powers
379     for (j in 1:k) {
380       M[[i,j]] <- MatrixPower(TempMatrix,j)
381     }
382     }
383   ## Finding alpha that gives the expected waiting time closer
384   #to the book value
385   alpha_min <- alpha[which.min(abs(TempExpTime-MMValue))]
386   ExpWaitTime <- TempExpTime[which.min(abs(TempExpTime-MMValue)
    )]
387
388   ## Return Values:
389   ## 1. alpha that gives the expected waiting time closer to
    the book value
390   ## 2. expected waiting time closer to the book value
391   ## 3. matrix powers of TempMatrix for each alpha
392   return(list(alpha_min=alpha_min, ExpWaitTime=ExpWaitTime,
    MatrixPower=M))
393 }
394   ## obtaining the alpha value that gives the expected waiting
    time
395   #closer to the matrix value ()
396 FuzzyResults = FuzzyExpWaitTime(labels = c(0,1), trials = 1000,
397               pattern = c(1,1,1,1), P =stock,
398               alpha <- seq(0.1,0.3,0.01), MMValue =22.64)
399
400 ## Print the alpha value that gives the expected waiting time
    closer
401 #to the book value
402 FuzzyResults$alpha_min
403 ## Expected waiting time closer to the book value
404 FuzzyResults$ExpWaitTime
405 ## Print the matrix power of 3 for 1st alpha value (i.e. alpha
    = 0.1)
406 FuzzyResults$MatrixPower[1,3]
407 ## Print the matrix power of 4 for 3rd alpha value (i.e. alpha
    = 0.3)
408 FuzzyResults$MatrixPower[3,4]
```

B.2 Chapter 3 codes

```
# Function to select cointegrated pairs

# data: a number of stocks to select cointegrated pairs
find_cointegrated_pairs <- function (data){
  n <- ncol (data)
  pvalue_matrix <- matrix(0, nrow=n, ncol=n)
  pairs <- list()
  m <- 1
  for (i in 1:n){
    for (j in 1:n){
      if(i>=j) {
        next;
      } else{
        S1 <- data[, i]
        S2 <- data[, j]
        result1 <- coint.test (as.numeric(S1), as.numeric(S2),
    output = FALSE)
        result2 <- coint.test (as.numeric(S2), as.numeric(S1),
    output = FALSE)
        pvalue_matrix [i, j] <- result1[,3][[1]]
        pvalue_matrix [j, i] <- result2[,3][[1]]
        if (result1[,3][[1]] < 0.05||result2[,3][[1]] < 0.05){
          pairs [[m]] <- c (i, j)
          m <- m+1}
      }
    }
  }
  newlist <- list(pvalue_matrix, pairs)
  return (newlist)
}
# function to calculate sign correlation
rho.cal<-function(X){
  rho.hat<-cor(sign(X-mean(X)), X-mean(X))
  return(rho.hat)
}
observed.vol <- function(X){
  X<-as.numeric(X)
  X.cdf <- ecdf(X)
  return(abs(X - mean(X))/(2*rho.cal(X)*sqrt(X.cdf(mean(X))*(1-
    X.cdf(mean(X))))))
}

#DD-EWMA
```

```
ddEW <- function(data){
  alpha<-seq(0.01, 0.5, 0.01)
  t <- length(data)
  cut.t <- 20 ### how many l values
  X.cdf <- ecdf(data)
  rho<- rho.cal(data)
  mu <- mean(data)
  vol<- abs(data - mu)/(2*rho*sqrt(X.cdf(mu)*(1-X.cdf(mu))))
  MSE_alpha <- rep(0, length(alpha))
  sn <- rep(0, length(alpha))
  for(a in 1:length(alpha)){
    s <- mean(vol[1:cut.t])
    error<-rep(0, t)
    for(i in 1:t){
      error[i]<-vol[i]-s
      s<-alpha[a]*vol[i]+(1-alpha[a])*s
    }
    MSE_alpha[a]<-mean(error[-(1:cut.t)]^2)
    sn[a] <- s
  }
  dd.vol.fore <- sn[which.min(MSE_alpha)]
  rmse <- sqrt(min(MSE_alpha))
  return(c(dd.vol.fore, rmse))
}
## function to estimate by Neuro volatility forecasts

neuro.vol <- function(data){
  data<-as.numeric(data)
  X.cdf <- ecdf(data)
  rho <- rho.cal(data)
  mu <- mean(data)
  vol <- abs(data - mu)/(2*rho*sqrt(X.cdf(mu)*(1-X.cdf(mu))))
  #vol.nnet <- list()
  vol.nnet <- forecast::nnetar(vol)
  nn.vol.fore <- forecast::forecast(vol.nnet, PI = TRUE, h=1)
  rmse <- sqrt(mean((nn.vol.fore$residuals)^2, na.rm=TRUE))
  return(c(as.numeric(nn.vol.fore$mean), rmse))
}

sr.train<-function(nu, vol, p){
  signals <- merge(nu, p*vol, -p*vol)
  colnames(signals) <- c("nu", "vol", "negvol")
  vec.sig<-ifelse((signals[1:length(index(signals))]$nu >
                   signals[1:length(index(signals))]$vol) &
                   (lag.xts(signals$nu, 1) < lag.xts(signals$vol
```

```
  , 1)), -1,
                ifelse((signals[1:length(index(signals))]$nu
                < signals[1:length(index(signals))]$negvol) &
                  (lag.xts(signals$nu, 1) > lag.xts(signals$
  negvol, 1)), 1, 0))

colnames(vec.sig) <- "vectorise.signals"
 # getting only the first signals
vec.sig[vec.sig == 0] <- NA # replace 0 by NA
vec.sig <- na.locf(vec.sig) # replace the missing values by
#last real observations
vec.sig <- diff(vec.sig)/2

# generate positions and calculate profit and loss
sim <- merge(lag.xts(vec.sig,1), beta[, 1], x[, 1], y)
colnames(sim) <- c("sig", "hedge", assets[1], assets[2])
sim$posX <- sim$sig * -100 * sim$hedge
sim$posY <- sim$sig * 100
sim$posX[sim$posX == 0] <- NA
sim$posX <- na.locf(sim$posX)
sim$posY[sim$posY == 0] <- NA
sim$posY <- na.locf(sim$posY)
pnlX <- sim$posX * diff(sim[, assets[1]])
pnlY <- sim$posY * diff(sim[, assets[2]])
pnl <- pnlX + pnlY
st_p <- sqrt(252)*mean(na.omit(pnl))/sd(na.omit(pnl))
return (st_p)}

# Load the required stocks
start.date = '2017-2-1' # starting date of stock
end.date = '2023-12-08' # ending date of stock
# Download the seleted stocks from Yahoo finance
getSymbols(c('SPY', 'ADBE','EBAY','MSFT','IBM', "GLD", "GDX",
  "EWA",
             "EWC", "IGE"),
           src = "yahoo", from = start.date, to = end.date)
stocks <- merge(SPY = SPY[, "SPY.Adjusted"], ADBE = ADBE[, "
 ADBE.Adjusted"],
                EBAY= EBAY[, "EBAY.Adjusted"], MSFT = MSFT[,
 "MSFT.Adjusted"],
                IBM = IBM[, "IBM.Adjusted"], GLD = GLD[, "GLD
 .Adjusted"],
                GDX = GDX[, "GDX.Adjusted"], EWA = EWA[, "EWA
 .Adjusted"],
                EWC = EWC[, "EWC.Adjusted"], IGE = IGE[, "IGE
```

```
    .Adjusted"])
124
125   ## Plot the stock prices and test for multiple cointegration
126   # selected two assets
127   assets <- c("SPY", "EWA")
128   pair.stock <- merge(stocks[, paste0(assets[1],".Adjusted")],
129                 stocks[, paste0(assets[2],".Adjusted")], join="
      inner")
130   colnames(pair.stock) <- assets
131   # # Plot the assets
132   plot(pair.stock, legend.loc=1)
133   # # Johansen test of cointegration
134   jotest=ca.jo(pair.stock, type="trace", K=2, ecdet="none",
      spec="longrun")
135   summary(jotest)
136   out<-cbind(jotest@teststat,jotest@cval)
137   colnames(out)<-c("test","10pct","5pct","1pct")
138
139 ##Rolling window
140   number<-0  ## change number from 0 to 17
141   rolling_size <- 90*number ##
142   stocks <- stocks[c((1+rolling_size):(180+rolling_size)),]
143
144   ##Algorithm 2
145   ####################
146   # Required functions to calculate p and profit
147   ######################
148   # Function to generate positions and calculate profit and
      loss for multiple
149   #stocks
150   PnL<-function(signals,nu,beta,x,y,assets){
151     len <- length(index(signals))
152     colnames(signals) <- c("nu", "volatility", "negvolatility")
153     vec.sig<-ifelse((signals[1:len]$nu > signals[1:len]$
      volatility) &
154               (lag.xts(signals$nu, 1) < lag.xts(signals$
      volatility, 1)), -1,
155               ifelse((signals[1:len]$nu < signals[1:len]$
      negvolatility) &
156               (lag.xts(signals$nu, 1) > lag.xts(signals$
      negvolatility, 1)), 1, 0))
157
158     colnames(vec.sig) <- "vectorise.signals"
159     # getting only the first signals
160     vec.sig[vec.sig == 0] <- NA # replace 0 by NA
```

```
161     vec.sig <- na.locf(vec.sig) # replace the missing values by
        last real
162     #observations
163     vec.sig <- diff(vec.sig)/2
164     # generate positions and calculate profit for two stocks
165     if(ncol(beta)==2){
166       sim <- merge(lag.xts(vec.sig,1), beta[, 1], x[, 1], y)
167       colnames(sim) <- c("sig", "hedge", assets[1], assets[2])
168       sim$posX <- sim$sig * -1000 * sim$hedge ##(step 7 of Algo
        2)
169       sim$posY <- sim$sig * 1000
170       sim$posX[sim$posX == 0] <- NA
171       sim$posX <- na.locf(sim$posX)
172       sim$posY[sim$posY == 0] <- NA
173       sim$posY <- na.locf(sim$posY)
174       PLX <- sim$posX * diff(sim[, assets[1]]) ##(step 9)
175       PLY <- sim$posY * diff(sim[, assets[2]]) ##(step 10 of
      Algo 2)
176       profit_loss <- PLX + PLY
177     }
178     # generate positions and calculate profit for three stocks
179     if(ncol(beta)==3){
180       sim <- merge(lag.xts(vec.sig,1), beta[, 1], beta[, 2], x
      [, 1],
181                    x[, 2], y)
182       colnames(sim) <- c("sig", "hedge1", "hedge2", assets[1],
      assets[2],
183                          assets[3])
184       sim$posX1 <- sim$sig * -1000 * sim$hedge1
185       sim$posX2 <- sim$sig * -1000 * sim$hedge2
186       sim$posY <- sim$sig * 1000
187       sim$posX1[sim$posX1 == 0] <- NA
188       sim$posX1 <- na.locf(sim$posX1)
189       sim$posX2[sim$posX2 == 0] <- NA
190       sim$posX2 <- na.locf(sim$posX2)
191       sim$posY[sim$posY == 0] <- NA
192       sim$posY <- na.locf(sim$posY)
193       PLX <- sim$posX1 * diff(sim[, assets[1]]) + sim$posX2 *
194       diff(sim[, assets[2]])
195       PLY <- sim$posY * diff(sim[, assets[3]])
196       profit_loss <- PLX + PLY
197     }
198     # generate positions and calculate profit for four stocks
199     if(ncol(beta)==4){
200       sim <- merge(lag.xts(vec.sig,1), beta[,1], beta[,2], beta
```

```
  [,3], x[,1], x[,2],
                  x[,3], y)
     colnames(sim) <- c("sig", "hedge1", "hedge2", "hedge3",
  assets[1],assets[2],
                       assets[3], assets[4])

     sim$posX1 <- sim$sig * -1000 * sim$hedge1
     sim$posX2 <- sim$sig * -1000 * sim$hedge2
     sim$posX3 <- sim$sig * -1000 * sim$hedge3
     sim$posY <- sim$sig * 1000
     sim$posX1[sim$posX1 == 0] <- NA
     sim$posX1 <- na.locf(sim$posX1)
     sim$posX2[sim$posX2 == 0] <- NA
     sim$posX2 <- na.locf(sim$posX2)
     sim$posX3[sim$posX3 == 0] <- NA
     sim$posX3 <- na.locf(sim$posX3)
     sim$posY[sim$posY == 0] <- NA
     sim$posY <- na.locf(sim$posY)
     PLX <- sim$posX1 * diff(sim[, assets[1]]) + sim$posX2 *
       diff(sim[, assets[2]]) + sim$posX3 * diff(sim[, assets
  [3]])

     PLY <- sim$posY * diff(sim[, assets[4]])
     profit_loss <- PLX + PLY
   }
   return(ProfitLoss=profit_loss)
}
# Functions to calculate the optimal value of threshold, $p$
SR.train<-function(nu, volatility, p, beta, x, y, assets){
   signals <- merge(nu, p*volatility, -p*volatility)
   colnames(signals) <- c("nu", "volatility", "negvolatility")
   # Implementation of profit
   profit.loss<- PnL(signals,nu,beta,x,y,assets)
   st_p <- sqrt(252)*mean(na.omit(profit.loss))/sd(na.omit(
  profit.loss))
   ##(step 13- annualized SR)
    return (st_p)
}
# Implementation of non-Gaussian filter algorithm
asset.pairs<-function(asset1, asset2){
   assets <- c(asset1, asset2) # selected two assets
   pair.stock <- merge(stocks[, paste0(asset1,".Adjusted")],
                       stocks[, paste0(asset2,".Adjusted")],
  join="inner")

```

```
241     colnames(pair.stock) <- assets
242    plot1 <- plot(pair.stock, legend.loc=2, main=paste(asset1," and", asset2))
243     ##Algorithm 1
244     # select pairwise cointegrated stocks
245     x <- pair.stock[, 1]
246     y <- pair.stock[, 2] ##higher price stock
247     x$intercept <- rep(1, nrow(x)) # create intercept
248     var_e <- 0.0001 # innovation covariance of observation
249     sigma_v <- var_e/(1-var_e)*diag(2) #covariance matrix of state
250     Ve <- 0.001
251     P_t <- 10^{-10}*diag(2)
252     P <- matrix(rep(0, 4), nrow=2)
253     I_t <- matrix(rep(0, 4), nrow=2) #information matrix
254     beta <- matrix(rep(0, nrow(y)*2), ncol=2)
255     y_fitted <- rep(0, nrow(y))
256     nu <- rep(0, nrow(y))
257     Q <- rep(0, nrow(y))
258
259     ############################
260     # Function to implement the Non-Gaussian maximum filter operations
261     ############################
262     #kalman_iteration <- function(y, x) {
263     for(i in 1:nrow(y)) {
264       if(i > 1) {
265         beta[i, ] <- beta[i-1, ] # state transition
266         P_t <- P + sigma_v # state covariance prediction (step 6)
267       }
268       y_fitted[i] <- x[i, ] %*% beta[i, ] # observation prediction (step 7)
269       Q[i] <- x[i, ] %*% P_t %*% t(x[i, ]) + Ve # observation variance prediction (step 10)
270       nu[i] <- y[i] - y_fitted[i] # prediction error ##(step 9)
271       K_gain <- P_t %*% t(x[i, ]) / Q[i] # information gain
272       # updating the state
273       beta[i, ] <- beta[i, ] + K_gain * nu[i] ## (step 11 of Algo 1)
274       I_t <- solve(P_t)+ t(x[i, ])%*%x[i, ] / Ve
275       P <- solve(I_t)
276       }
277     # Test of multiple cointegration
278     jotest=ca.jo(pair.stock, type="trace", K=2, ecdet="none",
```

```
    spec="longrun")
279  summary<-summary(jotest)
280  #res <- kalman_iteration(y,x) # Implementation of function
281  # Extract results
282  #beta <- xts(res[[1]], order.by=index(pair.stock))
283  plot2<-plot(beta[2:nrow(beta), 1],type='l',main =paste('
    Dynamic
284                    hedge ratio of',asset1, 'and', asset2),
    col = "blue")
285
286  plot3<-plot(beta[2:nrow(beta), 2],type='l',main =paste('
    Dynamic
287                      intercept of',asset1, 'and',asset2),col
     = "blue")
288
289  vol<-observed.vol(nu)
290  plot4<-plot(nu[3:length(nu)], type = "l", col = "blue",
291              main=paste("prediction error for", asset1, "and
    ", asset2))
292  plot5<-plot(vol[3:length(vol)], type = "l", col = "blue",
293              main=paste("volatility of prediction error for"
    ,asset1, "and",
294                         asset2))
295
296  vol.nu<-0; rmse.algo<-0
297  window <- 60
298  for(i in 1:(nrow(stocks)-window)){
299    result<-ddEW(nu[i:(window+i-1)])
300    vol.nu[i]<-result[1]
301    rmse.algo[i]<-result[2]
302  }
303
304  plot6<-plot(vol.nu, type = "l", ylab = "DD Volatility",
305              main=paste("DDEWMA Volatility for", asset1, "
    and", asset2))
306
307  #  trade signals
308  nu <- xts(nu, order.by=index(pair.stock))
309  sqrtQ <- xts(sqrt(Q), order.by=index(pair.stock))
310  sqrtQ[1:window]<-NA
311  vol <- xts(c(rep(NA, window), vol.nu), order.by=index(pair.
    stock))
312  volcombined <- merge(sqrtQ, vol)
313  colnames(volcombined) <- c("KFVEI", "DDVFI")
314  plot7<-plot(volcombined[(window+3):length(index(volcombined
```

```
   ))],
315        ylab='Volatlity', main = paste('DDVFI vs. KFVEI for',
   asset1, "and",
316        asset2), col=c('blue', 'red'), lwd=c(2,2))
317    legend("topright", legend = c("KFVEI", "DDVFI"), col = c("
   blue", "red"),
318           lty = 1, lwd = c(2, 2))
319    #Robust pairs trading strategy
320    p<-seq(0.75, 2, 0.01)
321    sr<-0
322    for(j in 1:length(p)){
323      sr[j] <- SR.train (nu,vol, p[j], beta, x, y, assets)
324    }
325    asr<- max(na.omit(sr))
326    plot8<- plot(p, sr, type = "l", col = "blue",main=paste("
   Annualized sharpe
327             ratio with DDEWMA vol for",asset1, "and", asset2))
328    p.opt.r<-p[which.max(sr)]
329    # create optimal trading signals
330    signals_ddvol <- merge(nu, p.opt.r*vol, -p.opt.r*vol)
331    colnames(signals_ddvol) <- c("nu", "vol", "negvol")
332    plot9<-plot(signals_ddvol[(window+3):length(index(signals_
   ddvol))],
333  ylab='nu', main = paste('Trading signals with DDEWMA vol for'
   , asset1, "and",
334  asset2), col=c('blue', 'red', 'red'), lwd=c(1,2,2))
335
336 ##Number of transactions
337    len.r <- length(index(signals_ddvol))
338    tradings.r<-ifelse((signals_ddvol[1:len.r]$nu > signals_
   ddvol[1:len.r]$vol) &
339       (lag.xts(signals_ddvol$nu, 1) < lag.xts(signals_ddvol$
   vol, 1)), -1,
340      ifelse((signals_ddvol[1:len.r]$nu < signals_ddvol[1:len.r
   ]$negvol) &
341      (lag.xts(signals_ddvol$nu, 1) > lag.xts(signals_ddvol$
   negvol, 1)), 1, 0))
342 trading.number.r = length(which(tradings.r == 1))+length(which(
   tradings.r == -1))
343 # # # Implementation of profit and loss fucntion to calculate
   cumulative
344    #profit
345    profit.loss.r<- PnL(signals_ddvol,nu,beta,x,y,assets)
346    op_profit.r<-sum (na.omit(profit.loss.r)) ##Optimal p
   profit
```

```
347     plot10<-plot(cumsum(na.omit(profit.loss.r)), main=paste("
    Cumulative profit,
348     $, for", asset1, "and", asset2, "using DDEWMA vol"), col =
    "blue")
349     ratio.r<- op_profit.r/trading.number.r
350 
351       #Traditional pairs trading strategy
352     # # ## Determinine optimal p to maximize Sharpe ratio
353     p<-seq(0.75, 2, 0.01)
354     sr.t<-0
355     for(j in 1:length(p)){
356       sr.t[j] <- SR.train (nu, sqrtQ , p[j], beta, x, y, assets
    )
357     }
358     asr.t<-max(na.omit(sr.t))
359     plot11<-plot(p, sr.t, type = "l", col = "blue",main="
    Annualized sharpe
360                  ratio with sqrtQ")
361     p.opt.t<-p[which.max(sr.t)]
362     # #   # create optimal trading signals
363     signals_trad <- merge(nu,  p.opt.t*sqrtQ, -p.opt.t*sqrtQ)
364     colnames(signals_trad) <- c("nu", "sqrtQ", "negsqrtQ")
365     plot12<-plot(signals_trad[(window+3):length(index(signals_
    trad))],
366     ylab='nu', main =paste("Trading signals using sqrt Q for",
    asset1,"and",
367     asset2), col=c('blue', 'red', 'red'), lwd=c(1,2,2))
368     ##Number of transactions
369     len.t <- length(index(signals_trad))
370     tradings.t<-ifelse((signals_trad[1:len.t]$nu > signals_trad
    [1:len.t]$sqrtQ) &
371           (lag.xts(signals_trad$nu, 1) < lag.xts(signals_trad$
    sqrtQ, 1)), -1,
372           ifelse((signals_trad[1:len.t]$nu < signals_trad[1:len
    .t]$negsqrtQ) &
373     (lag.xts(signals_trad$nu, 1) > lag.xts(signals_trad$
    negsqrtQ, 1)), 1, 0))
374 
375     trading.number.t = length(which(tradings.t == 1))+
376       length(which(tradings.t == -1))
377     # # # Implementation of profit and loss fucntion to
    calculate cumulative
378     #profit
379     profit.loss.t<- PnL(signals_trad,nu,beta,x,y,assets)
380     op_profit.t<-sum (na.omit(profit.loss.t)) ##Optimal p
```

```
    profit
381     plot13<-plot(cumsum(na.omit(profit.loss.t)), main=paste("
    Cumulative profit,
382             $, for", asset1, "and", asset2, "using sqrtQ"), col
     = "blue")
383      ratio.t<-op_profit.t/trading.number.t
384     return(c(plot1,plot2,summary,plot3,plot4,plot5,plot6,plot7,
    plot8,plot9,
385              plot10,plot11,plot12,plot13,p.opt.r,asr,op_profit.
    r,
386              trading.number.r,ratio.r, p.opt.t, asr.t, op_
    profit.t,
387              trading.number.t,ratio.t))
388   }
389   ## Before running below, you need to get the relevant data
     for windows
390   #by changing the window number in rolling window calculation
     (line 104)
391
392   ## Table II (with cointegration)
393   ##WIN 2 (win 0 cointegrated stocks)
394   a<-asset.pairs("EBAY","SPY")
395   b<-asset.pairs("MSFT","SPY")
396   c<-asset.pairs("EBAY","MSFT")
397   d<-asset.pairs("EWC","EBAY")
398   ##WIN 4 (win 0 cointegrated stocks)
399   e<-asset.pairs("EBAY","SPY")
400   f<-asset.pairs("MSFT","SPY")
401   g<-asset.pairs("EBAY","MSFT")
402   h<-asset.pairs("EWC","EBAY")
403    ##WINDOW 8
404    i<-asset.pairs("ADBE","SPY")
405   j<-asset.pairs("IBM","ADBE")
406   k<-asset.pairs("EWC","IBM")
407   l<-asset.pairs("EWC","ADBE")
408   m<-asset.pairs("IBM","SPY")
409   ##WINDOW 10
410   n<-asset.pairs("GDX","EBAY")
411   o<-asset.pairs("GDX","MSFT")
412   ##WINDOW 14
413   p<-asset.pairs("EWC","MSFT")
414   q<-asset.pairs("IGE","EWC")
415    #WINDOW 16
416   r<-asset.pairs("EWA","SPY")
417   s<-asset.pairs("EBAY","MSFT")
```

```
418   t<-asset.pairs("IGE","MSFT")
419   u<-asset.pairs("EWC","EBAY")
420   v<-asset.pairs("EWA","EWC")
421
422 ##%%%%%%%%%%%%%%%%%%%%
423 ##Networks and PageRank using emp correlation  for price series
424 # Load the required stocks
425
426 start.date = '2017-2-1' # starting date of stock
427 end.date = '2023-12-08' # ending date of stock
428 #Download the seleted stocks from Yahoo finance
429 getSymbols(c('SPY', 'ADBE','EBAY','MSFT','IBM', "GLD", "GDX", "
    EWA", "EWC", "IGE"),
430            src = "yahoo", from = start.date, to = end.date)
431 stocks <- merge(SPY = SPY[, "SPY.Adjusted"], ADBE = ADBE[, "
    ADBE.Adjusted"],
432                 EBAY= EBAY[, "EBAY.Adjusted"], MSFT = MSFT[, "
    MSFT.Adjusted"],
433                 IBM = IBM[, "IBM.Adjusted"], GLD = GLD[, "GLD.
    Adjusted"],
434                 GDX = GDX[, "GDX.Adjusted"], EWA = EWA[, "EWA.
    Adjusted"],
435                 EWC = EWC[, "EWC.Adjusted"], IGE = IGE[, "IGE.
    Adjusted"])
436
437 head(stocks)
438 tail(stocks)
439 nrow (stocks)
440 ## rolling window correlation
441 number<-0 ## change number from 0 to 17
442 rolling_size <- 90*number ##
443 stocks <- stocks[c((1+rolling_size):(180+rolling_size)),]
444 n<- nrow(stocks)
445
446 ## empirical correlation
447 Price =  stocks #(first output)
448 Price<-na.omit(as.matrix(Price))
449 cor_mat_emp<-cor(Price) ## 10x10 matrix
450 fivenum(abs(cor_mat_emp))
451 correlation.quan<-c()
452 quantile<- seq(0.01,1, by=0.001)
453 for (i in 1: length(quantile)){
454   correlation.quan[i]<-quantile(abs(cor_mat_emp), quantile[i])
455   }
456 df<- data.frame(cbind(quantile,correlation.quan))
```

```
plot(correlation.quan,quantile)
ggplot(df, aes(correlation.quan,quantile))+geom_line()+geom_point()+
  labs(x="Emp correlation", y="Quantile")
threshold <- seq(0.0,0.7,0.01)

library(igraph)
n<-length(threshold)
for(i in 1:n){
  new.correlation=abs(cor_mat_emp)
  new.correlation[new.correlation < threshold[i]]<-0
  plot(graph.adjacency(new.correlation,'undirected', mode =
  "upper",diag=F,weighted=T),main=paste("threshold =",threshold[i]))
  graph <- graph.adjacency(new.correlation,'undirected',
                           mode = "upper",diag=F,weighted=T)
  # Check if the graph is connected
  is_connected <- is.connected(graph)
   # Print the result
  if(is_connected) {
    print(paste(threshold[i],"The graph is connected."))
  } else {
    print(paste(threshold[i],"The graph is not connected.")) }

}

## set the correlation values below ... to zero to get rid of the weaker links
correlation.threshold.check<-abs(cor_mat_emp)
correlation.threshold.check[correlation.threshold.check < 0.53] <- 0
graph <- graph.adjacency(correlation.threshold.check,'undirected', mode = "upper",
                         diag=F, weighted=T)

# Set node names
V(graph )$name <- c('SPY', 'ADBE','EBAY','MSFT','IBM',
                    "GLD", "GDX", "EWA", "EWC", "IGE")
plot(graph,
     vertex.color = "lightgreen",
     vertex.size = 20,
     vertex.label.cex = 1.5,
     edge.color = ifelse(abs(E(graph)$weight) > 0.7, "red", "gray"),
     # Highlight edges with correlation > 0.7 edge.width =
```

```
    #abs(E(graph)$weight) * 5. Adjust edge width based on correlation strength
)

 ## pagerank
stochastic_matrix_emp <- (correlation.threshold.check) /
  rowSums(correlation.threshold.check)

# PageRank Algorithm
# Define PageRank function (p = probability of teleport)
PageRank <- function (A, p, output) {
  # Initialize transition matrix
  s <- matrix(rep(NA, ncol(A)), ncol = ncol(A))
  s[1, ] <- rep(1/ncol(A), ncol(A))
  i <- 1
  # Repeat Markov Chain until convergence
  while (T) {
    # Calculate transition vector at t + 1
    t <- rep(NA, ncol(A))
    for (j in 1:ncol(A)) {
      t[j] <- ifelse(sum(A[j, ]) == 0
                     , 1 / ncol(A)
                     , p / ncol(A) + (1-p) *
                       sum(A[, j] * (s[i, ] / apply(A, 1, sum))
    ))}
    s <- rbind(s, t)
    i <- i + 1
    # Break if converged
    if (i > 1) if (all(round(s[i - 1, ], 4) == round(s[i, ], 4)
    )) break
    }
  # Build and return output
  rank <- data.frame(as.character(1:ncol(s)), round(s[i, ], 4),
    rep(p, ncol(A)))
  colnames(rank) <- c('Node', 'PageRank', 'P')
  if (output) {
    cat(noquote('PageRank Output:\n\n'))
    print(rank[order(-rank$PageRank), c('Node', 'PageRank')],
   row.names = F)
    cat(noquote(paste('\nPageRank converged in', i, 'iterations
   .')))
  } else {
    return(rank[order(-rank$PageRank), ])
  }
}
```

```
A<-stochastic_matrix_emp
# Plot the graph
library(igraph)
# p=0.2
PageRank(A, .2, T)
# Effect of P(teleport) on ranking
library(ggplot2)
# Calculate permutations
df <- data.frame()
for (i in seq(0,1,.005)) {
  df <- rbind(df, PageRank(A, i, F))
}

names<-c('SPY', 'ADBE','EBAY','MSFT','IBM', "GLD", "GDX", "EWA"
    , "EWC", "IGE")
df$Node <- factor(df$Node, levels = 1:10, labels = names)

ggplot(data=df, aes(x=P, y=PageRank, group=Node, fill=Node,
    colour=Node)) +
  geom_line(linewidth=1) +
  scale_x_continuous(breaks = seq(0,1,0.1)) +
  expand_limits(y=0.135) +
  xlab('P(teleport)')
```

Bibliography

Avery, P. (1987). The analysis of intron data and their use in the detection of short signals. *Journal of Molecular Evolution 26*, 335–340.

Bakhach, A. M., E. P. Tsang, and V. Raju Chinthalapati (2018). Tsfdc: A trading strategy based on forecasting directional change. *Intelligent Systems in Accounting, Finance and Management 25*(3), 105–123.

Binyamini, H., R. Bitton, M. Inokuchi, T. Yagyu, Y. Elovici, and A. Shabtai (2021). A framework for modeling cyber attack techniques from security vulnerability descriptions. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2574–2583.

Bodjanova, S. (2005). Median value and median interval of a fuzzy number. *Information Sciences 172*(1-2), 73–89.

Bonanno, G., G. Caldarelli, F. Lillo, S. Micciche, N. Vandewalle, and R. N. Mantegna (2004). Networks of equities in financial markets. *The European Physical Journal B 38*, 363–371.

Bowala, S., J. Singh, A. Thavaneswaran, R. Thulasiram, and S. Mandal (2022). Comparison of fuzzy risk forecast intervals for cryptocurrencies. In *2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, pp. 1–8. IEEE.

Brunetti, M. and R. De Luca (2023). Pre-selection in cointegration-based pairs trading. *Statistical Methods & Applications 32*(5), 1611–1640.

Cartea, Á., S. Jaimungal, and J. Penalva (2015). *Algorithmic and high-frequency trading*. Cambridge University Press.

Chen, H., S. Chen, Z. Chen, and F. Li (2019). Empirical investigation of an equity pairs trading strategy. *Management Science 65*(1), 370–389.

Chi, K. T., J. Liu, and F. C. Lau (2010). A network perspective of the stock market. *Journal of Empirical Finance 17*(4), 659–667.

Dickey, D. and W. Fuller (1979, 06). Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association 74*.

Do, B. and R. Faff (2010). Does simple pairs trading still work? *Financial Analysts Journal 66*(4), 83–95.

Dobrow, R. P. (2016). *Introduction to stochastic processes with R*. John Wiley & Sons.

Engle, R. F. and C. W. J. Granger (1987). Co-integration and error correction: representation, estimation and testing. *Econometrica 55*, 251–276.

Finance, Y. (2020). Yahoo finance. retrieved from finance.yahoo.com. https://finance.yahoo.com/recent-quotes.

Flori, A. and D. Regoli (2021). Pairs-trading strategies with recurrent neural networks market predictions. In *Mathematical and Statistical Methods for Actuarial Sciences and Finance: eMAF2020*, pp. 217–222. Springer.

Freedman, D., R. Pisani, and R. Purves (2007). Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

Friedman, J. H. (1998). Data mining and statistics: What's the connection? *Computing Science and Statistics 29*(1), 3–9.

Gatev, E., W. N. Goetzmann, and K. G. Rouwenhorst (2006). Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies 19*(3), 797–827.

Gentleman, J. F. and R. C. Mullin (1989). The distribution of the frequency of occurrence of nucleotide subsequences, based on their overlap capability. *Biometrics*, 35–52.

Huck, N. (2013). The high sensitivity of pairs trading returns. *Applied Economics Letters 20*(14), 1301–1304.

Huck, N. and K. Afawubo (2015). Pairs trading and selection methods: is cointegration superior? *Applied Economics 47*(6), 599–613.

Johansen, S. (1991). Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica: journal of the Econometric Society*, 1551–1580.

Jothimani, D., C. Kavaklioğlu, and A. Başar (2018). Financial networks: A study of the toronto stock exchange. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4684–4691. IEEE.

Junyan, Z. and Y. Chenhui (2015). Sequence pattern mining based on markov chain. In *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 234–238. IEEE.

Kimou, K., B. Barry, M. Babri, S. Oumtanaga, and T. Kadjo (2010). An efficient analysis of honeypot data based on markov chain. *Journal of Applied Sciences 10*(3), 196–202.

Liang, Y., A. Thavaneswaran, and M. E. Hoque (2020). A novel algorithmic trading strategy using data-driven innovation volatility. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1107–1114. IEEE.

Liang, Y., A. Thavaneswaran, A. Paseka, W. Qiao, M. Ghahramani, and S. Bowala (2022). A novel optimal profit resilient filter pairs trading strategy

for cryptocurrencies. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1274–1279. IEEE.

Liang, Y., A. Thavaneswaran, N. Yu, M. E. Hoque, and R. K. Thulasiram (2020a). Dynamic data science applications in optimal profit algorithmic trading. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1314–1319. IEEE.

Liang, Y., A. Thavaneswaran, N. Yu, M. E. Hoque, and R. K. Thulasiram (2020b). Dynamic data science applications in optimal profit algorithmic trading. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1314–1319. IEEE.

Longmore, K. (2019). Kalman filter example: Pairs trading in r. robot wealth.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems 11*, 193–197.

Miao, G. J. (2014). High frequency and dynamic pairs trading based on statistical arbitrage using a two-stage correlation and cointegration approach. *International Journal of Economics and Finance 6*(3), 96–110.

Mota, M. R. (2023). Pairs trading: Using machine learning for the selection of pairs. https://medium.com/aimonks/pairs-trading-using-machine-learning-for-the-selection-of-pairs-24920cbed1b6.

O'Hara-Wild, M., R. Hyndman, and E. Wang (2024). *fabletools: Core Tools for Packages in the 'fable' Framework*. R package version 0.4.1, https://github.com/tidyverts/fabletools.

Sims, C. A. (1980). Macroeconomics and reality. *Econometrica*, 1–48.

Spedicato, G. A. (2017). Discrete time markov chains with R. *R J. 9*(2), 84.

Spitzner, L. (2003). *Honeypots: tracking hackers*, Volume 1. Addison-Wesley Reading.

Stoffer, D. and M. D. Stoffer (2023). Package 'astsa'. *Blood 8*, 1.

Sun, Y., S. Zhu, Y. Zhao, and P. Sun (2022). A user-friendly two-factor authentication method against real-time phishing attacks. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pp. 91–99. IEEE.

Thavaneswaran, A., S. S. Appadoo, and A. Paseka (2009). Weighted possibilistic moments of fuzzy numbers with applications to garch modeling and option pricing. *Mathematical and Computer Modelling 49*(1-2), 352–368.

Thavaneswaran, A., Y. Liang, S. Bowala, A. Paseka, and M. Ghahramani (2022a). Deep learning predictions for cryptocurrencies. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1280–1285. IEEE.

Thavaneswaran, A., Y. Liang, S. Bowala, A. Paseka, and M. Ghahramani (2022b). Deep learning predictions for cryptocurrencies. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1280–1285. IEEE.

Thavaneswaran, A., Y. Liang, Z. Zhu, and R. K. Thulasiram (2020). Novel data-driven fuzzy algorithmic volatility forecasting models with applications to algorithmic trading. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8. IEEE.

Thavaneswaran, A., A. Paseka, and J. Frank (2020). Generalized value at risk forecasting. *Communications in Statistics-Theory and Methods 49*(20), 4988–4995.

Thavaneswaran, A., K. Thiagarajah, and S. S. Appadoo (2007). Fuzzy coefficient volatility (fcv) models with applications. *Mathematical and Computer Modelling 45*(7-8), 777–786.

Thavaneswaran, A. and M. Thompson (2019). Nonnormal filtering via estimating functions. In *Advances on Theoretical and Methodological Aspects of Probability and Statistics*, pp. 173–183. CRC Press.

Xinming, O., S. Govindavajhala, and A. Appel (2005). A logic-based network security analyzer. In *14th USENIX Security Symposium; http://www. usenix. org/events/sec05/tech/ou. html: Baltimore, Maryland, USA*.

Yang, A., W. Zhang, J. Wang, K. Yang, Y. Han, and L. Zhang (2020). Review on the application of machine learning algorithms in the sequence data mining of dna. *Frontiers in Bioengineering and Biotechnology 8*, 1032.

Yang, C., Y. Chen, L. Niu, and Q. Li (2014). Cointegration analysis and influence rank—a network approach to global stock markets. *Physica A: Statistical Mechanics and its Applications 400*, 168–185.

Ying, W. (2006). Nad+ and nadh in cellular functions and cell death. *Frontiers in Bioscience-Landmark 11*(3), 3129–3148.

Zhu, Z., A. Thavaneswaran, A. Paseka, J. Frank, and R. Thulasiram (2020). Portfolio optimization using a novel data-driven ewma covariance model with big data. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1308–1313. IEEE.