VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Bachelors thesis

The Visualization of Regularities and Coincidences of Stock Time Series in a Mobile Application

Done by:
Vytautas Kondratas signature

Supervisor:
Ph. D. Tadas Meškauskas

Vilnius
2020

Contents

Abstract

The aim of this project was to create a mobile application capable of using stock time series to generate recurrence plots and analyse them to determine signal stationarity. The application features functionality for finding, storing and observing information about different stocks and generating custom colorless and colored recurrence plots using stock time series intervals. The generated diagrams can be analysed to calculate relative stock time series stationarity.

Several analyses were performed. A study was conducted to determine relative stock time series stationarity by analysing generated recurrence plots. This was achieved by dividing the images into smaller parts and calculating the local pixel densities of each such part. These were then compared to the local pixel densities of stationary and non-stationary recurrence plots to calculate a relative percentage value. Other studies performed include comparing observations in recurrence plot images created with different delay embedding parameters and comparing the performance of different vector distance calculation norms. It was determined that the Maximum norm had the best performance when generating recurrence plots.

Santrauka

Akcijų biržos kainų atsitiktinumų ir dėsningumų vizualizavimas mobilioje aplikacijoje

Rašto darbo tikslas buvo sukurti mobilią programėlę akcijų biržos kainų kitimo stebėjimui ir rekurentinių diagramų braižymui.

Sukurta programėlė turi funkcionalumą leidžiantį vartotojui rasti ir peržiūrėti akcijų kainas. Akcijos gali būti išsaugomos vartotojo sukurtuose portfeliuose. Įgyvendintas rekurentinių diagramų braižymas naudojant pasirinktų akcijų laiko eilutes, bei algoritmo slenkstinio atsumo kintamojo automatinis nustatymas. Įvertinus diagramų piešimo naršumą, naudojant didelius kiekius duomenų, įgyvendintas duomenų praretinimas. Programėlėje, taip pat įgyvendintas akcijos kainos laiko eilutės stacionarumo įvertinimas, naudojant sugeneruotas rekurentines diagramas. Stacionarumo įvertinimas suteikia vartotojui informacijos apie reliatyvų akcijos kainos pokyčio pastovumą.

Buvo atlikti keli tyrimai. Pirmiausia buvo atlikta rekurentinių diagramų analizė siekiant apskaičiuoti reliatyvų signalo stacionarumą. Tai padaryta apskaičiuojant lokalius mažesnių kvadratėlių diagramoje tankius ir lyginant juos su aukšto bei žemo stacionarumo signalų rekurentinėmis diagramomis. Tam atlikti buvo įvykdytas mažiausio stacionarumo nustatymas naudojant didelį kiekį įvairių akcijų rekurentinių diagramų. Taip pat buvo palygintos rekurentinės diagramos naudojant skirtingus laiko vėlinimo parametrus. Nustatyta, kad kai pasirenkamos reliatyviai didelės šių parametrų reikšmes, rekurentinės diagramos tampa neinformatyvios. Taip pat buvo palyginta skritingų slenkstinio atsumo apsakaičiavimo būdų įtaka diagramų piešimo laikui. Nustatyta, kad trumpiausias diagramos piešimo laikas yra pasiekiamas naudojant maksimumo metriką.

Introduction

Financial data analysis has been a topic of interest in the economics community ever since first instances of such data were collected. Today, more data is generated and stored than ever before, spanning multiple financial areas. A great example of this is the ever increasing repositories of historical stock time series. Each second, the price fluctuations of stocks are recorded to be viewed and examined by interested parties. Observing such data can provide insights into the history of the time series. However no amount of historical data observation can be used to predict fluctuations of such data in the future. Financial data is chaotic and dependant on too many external variables to be accurately predicted.

Still, various tools have been developed to detect and display recurrences and coincidences present in financial data. One of these is the recurrence plot. This method of data analysis considers the changes between the states of a data signal and portrays this information as an image. The image can be used to determine statistical properties of a signal, such as stationarity, periodicity, trends. These can provide additional insight into a the price fluctuations of a stock and is a valuable feature in a system designed to observe stock prices changes.

An application with such functionality can provide the user with insights into the statistical properties of a stock time series and thus help him determine whether a stock is worth investing in.

The goal of this project was to create a mobile application capable of using stock time series to generate recurrence plots and analyse them to determine signal stationarity.

To complete the goal of the project, several tasks were accomplished:

1. Use stock time series to generate recurrence plots;
2. Analyse recurrence plots to determine relative stationarity of a stock time series;
3. Investigate and compare the use of different vector distance norms and delay embedding parameters when generating recurrence plots.

Some of the stock observation features and their implementation were examined in the authors previous course work [18]. These along with any previous issues addressed can be found in section 2.5. When faced with performance issues while generating recurrence plots with a large range of values, data downsampling was implemented (see section 2.7). A variable of the recurrence plot algorithm, threshold distance, was optimized to be calibrated automatically without input of the user. (see section 2.8). Colorless recurrence plots were analysed to determine the relative stationarity of a stock time series (see section 3.1), which provides the user with insight into the volatility of a stock.

The project contains abstracts in English and Lithuanian. The first part of the project focuses on the theoretical part of the implementation, with information about financial data, its analysis and generation of recurrence plots. The second section focuses on the creation of the mobile application and implementation of its features. The final section contains studies performed using recurrence plots.

1 Data and Methods

First, this section will describe the types of data that can and will be used when drawing recurrence plots. Later, we will proceed to investigate alternative methods for data recurrence analysis. Lastly, recurrence plots will be explored along with their application when analysing time series.

1.1 Time series

In order to start visualizing features of information, we must consider the data itself. To search for recurrence a large stream of data is required, where values with historical information can be compared and analysed. Such data could be represented by an arbitrary data signal (see section 1.3) or a time series.

A time series is formally a set of measurements that have been recorded at specific points in time. The measurements are usually arranged in a chronological order. A time series is considered to be *discrete* when observations are made at distinct separate points in time. *Continuous* time series is one where measurements are made continuously over some time interval.

Time series usually specifies a change in state of some specific variable or process. It can describe mechanisms of varied complexity from the simple movements of a pendulum to meteorological shifts or even price fluctuations of the stock market. Time series data is frequently used in domains of science, engineering, economics and sociology. [23]

Time series can be classified as stationary or non-stationary. *Stationarity* in time series means that the statistical properties of a processes being described do not change as time passes. In other words, stationarity suggests that while the process itself describes change, the way this change happens remains constant. [21]

Trends in time series describe continuous increases or decreases of the value of a variable. *Seasonality* describes periodic short term changes that can be linked to seasonal and calendar related effects. These features can sometimes be recognised, when observing a visual representation of a time series, such as a line chart.

1.2 Stocks

Financial data can be considered to be any data that describes metrics or assets related to finance. A few examples of such metrics include companies' financial indicators, crypto and state-backed currency price fluctuations, stock and equity share prices and volumes sold. These metrics are often described as time series. A good example of this data, and one that we will use when analysing time series recurrence, is the stock.

A stock (or share) is a divided portion of the ownership of a company. Simply put, when you own a stock of a particular firm, you own a part of the company itself. Stocks are typically separated into two types: common stock and preferred stock. A *common stock* typically gives the owner a voting right that can be used when making corporate decisions. A *preferred* stock usually carry no such rights, however the owner is entitled to a payment of a fixed divided. [15]

Stocks can be bought and sold by their shareholders. This is usually done at specialized organizations called stock exchanges. These institutions provide a means to trade stocks, bonds and other securities as well as redeem earned dividends. Many different stock exchanges exist varying by region and popularity [16]. The oldest and largest is the New York Stock Exchange, which contains roughly 40% of the worlds stock market capitalization. Second is popularity is the NAS-

DAQ Stock Exchange. While not as large as its predecessor, NASDAQ contains stocks of the large fast growing technology companies such as Google, Apple and Microsoft. Stock data used in this project will include stocks from these exchanges.

Due to constant trading the prices of stocks continuously fluctuate. Similar to other commodities, stocks are subject to supply and demand. Many factors come into play when determining a price of a stock, such as the value of companies earnings, news about the company or related domains, trends in science and technology and so on. This constant fluctuation makes stock prices very volatile and difficult (or near impossible) to predict [12]. Stock prices are constantly recorded by interested parties. Historical price data can be used to observe a companies financial history, as well as indicate future growth or lack of it.

Stock data is often visualized in charts portraying price over some period of time. The most widely used to way to visualize this information is the open-high-low-close chart.

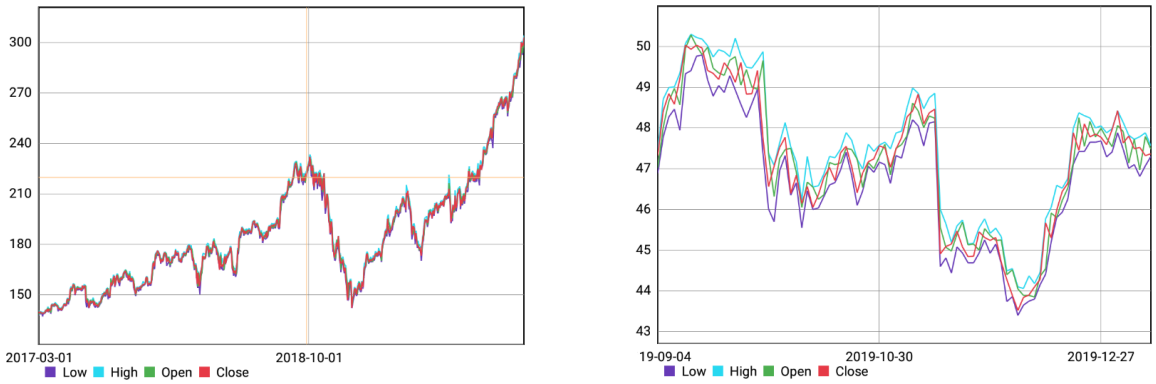


Figure 1. Open-high-low-close charts: Apple stock over 2 year period (on the right) and Cisco stock over 3 month period (on the left). Data Source: Alpha Vantage [10].

Figure 1 shows an open-high-low-close charts of two stocks. Each vertical structure portrays a summary of price changes within a specific time period (in this case - a day). The lines show the range of the price and are recorded in four values. These values are referred to as *open* (for opening price), *high* (for maximum price), *low* (for minimum price) and *close* (for closing price) [14].

1.3 Other data

Stock metrics provide great factual data for analysing time series. However, as mentioned in 1.2 stock prices are a volatile and unpredictable medium. Stocks cannot be counted on to provide certain results for verifying the correct execution of methods to be used for data recurrence analysis. Thus, predictable data is also required.

If a data signal has strong static, trend or periodicity properties, these would be detectable by data analysis methods and could be used to confirm their validity. Examples of these data sets include mathematical expressions, such as a sine wave.

An opposite approach may also be applied. Time series that contain excessive noise or purposely randomly generated values, when applied to our methods, should show no defined static, trendy or periodic qualities.

Figure 2 shows examples of periodic and chaotic data. The sine wave data set displays strong stationarity and periodicity, while the randomly generated value chart displays no discernible patterns, properties which are detectable by using recurrence plots (see 1.5.5).

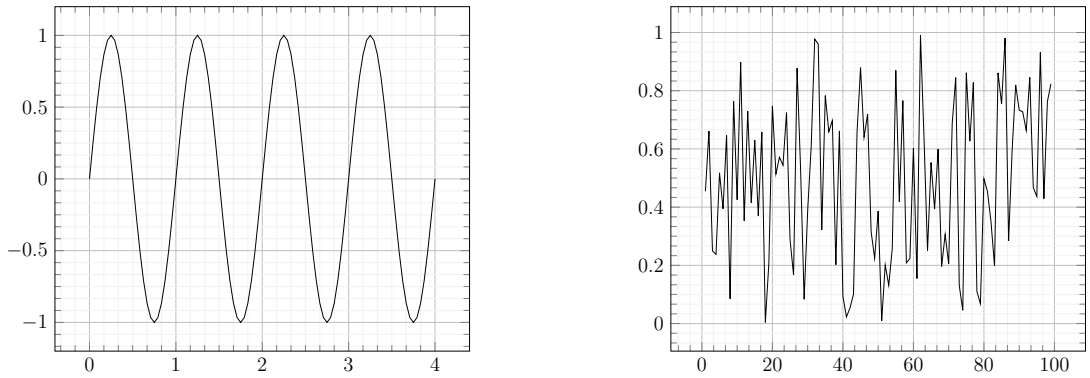


Figure 2. Sine wave signal (on the left) and randomly generated signal (on the right).

1.4 Related methods

This project will focus on visualising data recurrence by use of recurrence plots (see 1.5). However other methods of pattern discovery and time series forecasting exist. Tools like spectral analysis and time series regression models will be passingly explored in this section.

The first, *spectral analysis* is a widely used data analysis method. The idea of this approach is transforming a time series into high and low frequency signals. That is achieved by comparing the time series with sine and cosine functions of different frequencies. By summing up sine and cosine functions with different amplitudes, an artificial time series resembling the original can be constructed. The decomposition of a time series to these harmonic functions is called a *Fourier* (or spectral) decomposition. It can be used to research statistic properties by analysing the mathematical expressions behind the time series. [22]

Another way to analyse time series is to use *regression analysis*. Regression analysis describes mechanisms used to detect relationships of independent variables (or functions) that describe a process and values that they produce (also know as dependent variables). One such method is called *linear regression*, which focuses on visualizing said relationships as straight lines. *Simple* linear regression is used when only a single independent variable is present. *Multiple* linear regression is used when there are multiple independent variables. These established relationships can be used to predict future values of dependant variables. [24]

Auto regressive (or "AR") model is another way to forecast time series. It takes into account statistically significant past data values to model a function that can identify upcoming values in the time series. Another related model is the *moving average* model (or "MA") which describes determining and adjusting future predictions based on past prediction error rates. A combination of these models is the autoregressive–moving-average (or "ARMA") model which takes into account both past values and past prediction errors of a time series. All of these models require the time series to be stationary. When analysing a trending or seasonal time series, autoregressive-integrated–moving-average (or "ARIMA") model can be used, which, instead of predicting the time series itself, predicts differences between values of sequential time stamps. [23]

1.5 Recurrence plots

It is not difficult to conceive that the value variations of a financial metric may sometimes be produced in a recurrent manner. Certain patterns sometimes emerge in the fluctuations of signal values that, when observed, provide additional insights into the metric in question. For example, a time series might have trends, recurring increases or decreases in value and other discernible

properties. The opposite is also possible, the values might show no recurrence based on its change and the series might even seem randomly generated.

Such recurrence or lack of it may be observed by using data to generate recurrence plots, which are used in the field of non linear data modeling.

1.5.1 Phase space trajectories

Before we dive into the definition of recurrence plot, we must first familiarize with the concept of phase space trajectories.

It is indisputable that natural and mechanical systems change over time. From weather fluctuations to stock market price changes, these systems display dynamic behaviour that depends on numerous distinct variables. However the comparing values of the signal is not the correct approach when trying to detect recurrence between states. Two consecutive signal values being equal does not translate to the next values being equal as well. In order to compare changes of states within a signal, *signal states* must be compared [19].

Signal states represent the state a variable at a specific point in time. Signal states are commonly depicted as vectors (or trajectories) in phase space. Phase space is a space describing all possible states of a variable within a system. For a linear signal, signal states can be described as a pair of two consecutive signals. However, as variables of a system sometimes depend on more than the relationship of only two system variables, the signal state is usually described as a D-dimensional vector, that includes D different components. The relationship between D at a certain point in time *t* can be expressed as a vector

$$\vec{x}(t) = (x_1(t), x_2(t), ..., x_D(t)) \tag{1.1}$$

that represents a state of a variable in the *D*-dimensional phase space. [20]

When applying a dimension variable to signal states, values are grouped together using a *time delay* parameter d. This can be used to provide a more complete definition of a phase space vector

$$\vec{x}(t) = (x(t), x(t+d), x(t+2d), ..., x(t+(D-1)d)), t = 0, 1, ..., M, M = N-(d-1)d \tag{1.2}$$

where t is the time index of a value in a signal. According to the literature [19] $D = 2$ and $d = 1$ are most commonly used when generating recurrence plots.

1.5.2 Definition of the matrix

Recurrence plot (or "RP") is a tool that displays recurrences of phase space trajectories expressed as a two-dimensional matrix of ones and zeros.

The formal definition of the matrix is

$$R_{i,j}(\varepsilon) = \mathcal{H}(\varepsilon - ||\vec{x}_i - \vec{x}_j||) \tag{1.3}$$

where $i, j = 1, 2, 3, ..., N$ and denotes the number of signal states $\vec{x}_{i,j}$ measured, ε is a threshold distance (see 1.5.3), $\mathcal{H}$ is something called a Heaviside function and $||.||$ is a norm used to calculate distance between two states (see 1.5.4) [19].

The Heaviside function in an expression

$$\mathcal{H}(x) = \begin{cases} 0 \text{ when } x < 0 \\ 1 \text{ when } x \geq 0 \end{cases} \tag{1.4}$$

which describes the elements of the matrix [17].

Simply put, the definition of the matrix is a comparison of distance between different phase space trajectories with a chosen threshold distance. If distance between to states is lower than the threshold, the number in the matrix with coordinates (*i*,*j*) is a **zero**, otherwise the number is a **one**. Both axes of the matrix represent time.

In practice an RP is often represented by a bitmap image with black and white dots that correspond to ones and zeros of the matrix. The image portrays structures that provide insight into the data series in question. If data is a time series, these insights include stationarity, periodicity, trends and whether the series displays constant or near constant behavior. [20] [9] The structures and their interpretation will be further explored in section 1.5.5.

1.5.3 Threshold distance

One of the most critical parameters of a recurrence plot is the threshold ε. It is a user chosen scalar value that tells us whether the distance between two vectors is sufficient for the system states to be considered recurrent.

The threshold must be chosen with care, as an improper selection may result in misleading recurrence plots. Relatively high threshold values lead to more signal states being considered recurrent, thus resulting in more pronounced structures within an RP, which can give an impression of recurrence in its absence. The opposite is also possible. Choosing an excessively low threshold leads to undeveloped or imperceptible structures in a plot. Another thing to consider is noise within time series of signal states. If too much noise is present, a higher threshold is required or no structures will form in an RP. [20]

Some general rules for choosing a correct threshold have been suggested. According to the report [20], a threshold should not exceed 10% of mean or the maximum phase space diameter. The diameter can be calculated by taking vectors representing the minimum and maximum values of the data. The distance between these two vectors is the diameter.

A way to manually choose a correct threshold is to select a random value and create a RP with it. Then, adjust the threshold by doubling and halving it to create subsequent plots. These should then be compared to see which thresholds provide most adequate representations. This process is repeated until a desired coverage of black pixels is reached. [19]

1.5.4 Distance calculation

The distance between two D-dimensional vectors representing phase space trajectories can be calculated by using functions called norms (or metrics).

The first is the l_1 metric or *Manhattan norm*

$$||v|| = |v_1| + |v_2| + |v_3| + ... + |v_D| \tag{1.5}$$

which is a sum of absolute vector values.

The second is the l_2 metric or *Euclidean norm*

$$||v|| = \sqrt{v_1^2 + v_2^2 + ... + v_D^2} \tag{1.6}$$

which calculates the square roots of sums of squares.

The third is the l_∞ metric or *Maximum norm*

$$||v|| = max(|v_1|, |v_2|, |v_3|, ..., |v_D|) \tag{1.7}$$

which calculates the maximum vector value.

In order use norms to calculate distance between vectors, $||v||$ should be changed to the difference $x_i - x_j$ of signal states $x_{i,j}$. [19]

1.5.5 Interpretation

According to an online recurrence plot reference [9], structures present in recurrence plots can be characterized into four typologies:

- Homogeneous - evenly distributed pixels typically signifying the system is stationary;
- Periodic - structures parallel to the central diagonal and checkerboard patterns through out the plot;
- Drift - density of black pixel fading towards the corners of the image. Signifies an underlying trend with slowly changing parameters;
- Disrupted - white areas or bands in the plot. Signifies abrupt changes in the time series.

Smaller structures can also be observed when examining a plot more closely. These can be categorized into a few main groups:

- small individual points, that occur when recurrence between signal states is rare or heavy fluctuations are present;
- diagonal lines, that happen when a state visits roughly similar states at varying times;
- vertical (horizontal) lines, which occur when a state does not change or change very slowly over a period of time.

2 Creation of the mobile application

The aim of the project is to create a mobile application capable of presenting the user with information about stocks. Many such applications already exist in repositories such as "Google Play" and Apple's "App Store", all with similar functionality. These can be investigated to define and implement core functionality of a stock viewing application with additional features of recurrence plot generation.

2.1 Similar system analysis

To investigate the most prominent features of a stock viewing application, a similar system analysis was required to determine the functionality and design of such a system. Such an analysis was performed in the authors course work [18]. According to the analysis, most prominent features include searching for stocks and displaying their most recently updated quotas, ability to create and manage stock folders (portfolios) and manual refreshing information to receive the updated data.

It is also important to note, that none of the analysed applications featured functionality for generating and analysing recurrence plots. In fact, searching the "Play store" for any application capable of generating these images turned up no results. This reaffirms the decision to create a mobile application for stock application with recurrence plot generation ability as current selection of such applications is non existent.

2.2 Tools and architectural decisions

The application created was designed for use on the Android operating system as it is supported by a wide range of mobile devices and thus the application can reach the widest audience of users.

The development of the application was largely done using a free open source android development tool "Android Studio" which provides the user with a visual layout editor, built-in emulator with a wide selection of android devices and a flexible build system powered by Gradle. [1]

The application use two data sources for fetching information: Alpha Vantage API [10] for receiving daily, monthly and weekly stock quotas and time series and Exchange Rates API [3] for receiving latest exchange rates for the currency converter.

The source code was written in Java, version 8, as it contains functions and streams, which are useful when processing data. Most of the recurrence plot functionality was implemented using default libraries in Android SDK, however some additional libraries were also used. These are as follows: *Material Search View* [7] for implementing stock search functionality, *FasterXML Jackson* [4] for parsing Json responses, *MP Android Chart* [8] for generating graphs using stock time series, *Material Design* [11] resources were used to implement application user interface and *LoopJ Android Async Http* [6] for performing HTTP request to the server.

2.3 Project structure

The application was implemented using the MVC (Model-view-controller) design pattern. It is a commonly used design pattern for developing web and mobile applications. The pattern [13] divides the logic of a program into three components. The model represents data structures used in the application. The view represents the visual interface as seen by the user. The controller contains program logic that, upon a request, generates a response using the model and displays it to the user in a view. Figure 3 shows a diagram of the MVC design pattern.

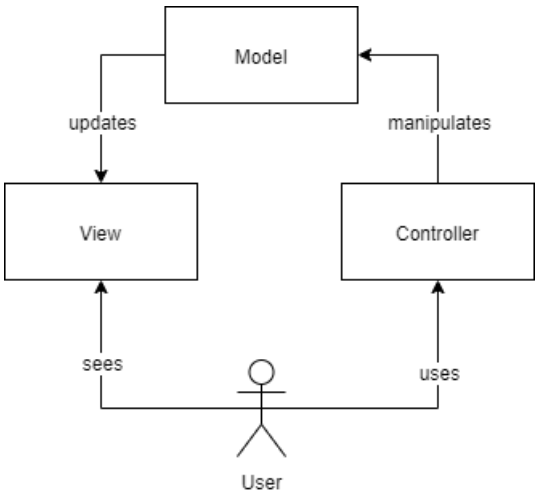


Figure 3. MVC design pattern.

This project structure was used to store application source code and resources:

- java/mobilestocks/
  - controller/
  - core/

        * recurrence_plot/
        * statistical_analysis/
    - exception/
    - model/
    - util/
- res/
    - layout/

The main application functionality is stored in *java/* directory. Package *core/* contains recurrence plot generation and analysis features. Package *controller/* contains the controllers of applications views. Package *exception/* contains custom exceptions. Package *model/* contains models used in the system. Package *utils/* contains utilities such as http request generation, JSON parsing and constants. Directory *res/layout/* contains the layouts of views used in the application.

2.4 Device specifications

Two devices were used in the making of the project: a mobile phone to test the implementation and execution of the application and a laptop computer to execute tasks requiring more computing power.

The mobile phone is Huawei P30. Device specifications [5] include an octa-core (2x2.6 GHz & 2x1.92 GHz & 4x1.8 GHz) processor, 8Gb of RAM and a 6.1 inch display.

The laptop computer is Lenovo Yoga C940 and features a Intel Core i7 processor running at 3.9 Ghz, 16Gb of dedicated RAM and 1TB of solid state memory.

2.5 Stock viewing application functionality implementation

This section contains features specific to a stock viewing application (as discussed in section 2.1. The implementation of these features was initially examined in the authors previous work [18] and is presented along with any issues that have been since noticed and addressed.

2.5.1 Searching for stocks

The search was implemented in the first window initially started when the user enters the application. It is reachable by using the navigation button with a search indicator on the bottom of the application interface (see appendice C).

Material Search View [7] library and its resources were used to create a clickable Search View layout in the top toolbar of the application. When clicked, the user is prompted to enter a query to proceed. When a query is submitted, it is passed to a method, that uses it to generate a search URL present in Alpha Vantage API documentation [10]. The generated link is is then used to make a HTTP Get request to the server which, on success, provides a JSON response with the search results. The response is parsed with a parsed made using *FastXML Jackson* [4] library and used to create a list of search results, that are then presented to the user. When a search result is selected, the user is taken to the detailed stock view (see section 2.5.3).

2.5.2 Stock folder creation and management

The stock folder (or "portfolio") was implemented in a different window that is reachable by using the navigation button on the bottom of the application interface (see appendices). Portfolios are divided into two parts, the portfolio list view (appendice D) and a portfolio item view (appendice E).

The user can observe their entire library of portfolios in the portfolio list view and create new portfolios. When a folder item is selected, the user is taken to a different view, containing the stocks of a selected portfolio. Stocks can be added to the portfolio by selecting the appropriate button, which takes the user to the search view. Stocks can be removed by pressing and holding on a specific stock, until a prompt appears. The portfolio name can be edited and the portfolio itself deleted by using the appropriately stylized buttons in the view.

Portfolio items are instances of a *Portfolio* class, which are wrapped within a PortfolioList class. As the list or its items get modified, changes are recorded and saved as Java persistent object in application data directory within the device file system.

2.5.3 Detailed stock view

Whenever a stock is selected from either the search view of a portfolio, the user is taken to a view (appendice F) that contains detailed information about the stock. This view has been modeled by considering the designs of similar applications observed in the authors course work [18].

The information presented includes stock symbol, currency, latest change time, percentage, value and most recent stock prices (open, high, low, close). The view also contains a graph featuring stock price changes over different periods of time.

When first entering the view, the stock symbol is used to generate queries to Alpha Vantage API [10] that, when executed, fetch compact intraday and full length daily stock price time series along with stock quote information. When all queries are performed successfully, the resulting data is used to create a stock object that is then saved to device memory in a stock data file. To reduce the amount of HTTP Get request performed, upon revisiting the stock view of a particular stock symbol, the data file is queried and, if the symbol data is present, the information is loaded into the view. Stock data can be updated by pressing the refresh button.

As mentioned before, the view contains a line graph of stock price changes over a selected period of time. The graphs are generated when any of the corresponding time range buttons are pressed. The generation of graphs is achieved by converting the desired stock time series into Line Data set objects, that are added to a ChartView using the *MP Android Chart* library [8].

2.5.4 Currency converter

An additional feature not present in any of the stock viewing applications examined is the currency converter. As stocks each have different currencies, a tool to quickly convert between them is a good practical feature. The user can navigate to the converter by using the navigation button on the bottom of the interface.

The converter view (appendice K) features two input fields and spinners (input fields where the user selects an option form a list) to provide the application with required input and output currencies and prices. When all the data is entered, the convert button can be pressed to perform the conversion. Upon click, the input currency is used to generate a URL to query the *Exchange*

Rates API [3] and receive a JSON response containing conversion ratios to other currencies. The exchange rate of the desired output currency is then found and used to calculate the resulting value.

The currency calculator is also reachable from the detailed stock view by pressing the designated button near the stock currency value. If pressed, the converter is opened and the stock price and currency are entered into the input fields.

2.6 Recurrence plot generation

2.6.1 Colorless plot generation

The most commonly used recurrence plot is a diagram consisting of black and white pixels. The structures in such an image are pronounced and easily discernible by an observer.

When generation of a plot is initiated, the selected range of data is extracted from the entire stock time series provided by the stock view. The data is used to construct signal state vectors using the time delay and dimension parameters supplied by the user. The first step is calculating the adjusted length of the signal using the formula

$$adjusted\_length = signal\_size - (dimension\_param - 1) \cdot time\_delay\_param \quad (2.1)$$

where $signal\_size$ is the amount of values present in the time series. The length represents the amount of signal states present at the end of the construction. Then, with the help of a loop that iterates times equal to the adjusted signal length, the data is constructed into a list of signal states which are later used to draw the plot.

The plot itself is an object of a Bitmap class in the default Android SDK. When all signal states have been constructed, a Bitmap is created using the adjusted signal length as dimensions for the image. All signal states are then compared with one another to check whether the norm of the two vectors is within the provided threshold distance (see section 1.5.3). If two vectors are within the desired threshold, a black pixel is placed at the coordinates which are represented by the indexes of the two vectors. Otherwise, the pixel drawn is white. This process is repeated until all vectors have been compared and the resulting image is returned as a Bitmap object.

2.6.2 Colored plot generation

A less frequently used method for visualizing recurrence plot is the colored recurrence plot. Generated with colors from a custom color scale, the plot provides insight into how recurrence plots with different threshold distances compare with one another.

The process of creating such a recurrence plot is similar to that of a colorless one and starts with constructing the time series data into signal states. Then the amplitude of time series values is calculated by subtracting the maximum and minimum values of the series. The resulting value is a distance, which represents a threshold that would generate a completely blank recurrence plot (as no two signal states would ever be within this threshold).

An array of colors was chosen to represent a colored scale. The amplitude is divided into scalar values representing different threshold distances which are equal in number to the amount of colors in the scale. Then a bitmap image is created using the adjusted signal length as image dimensions. The array of colors is iterated over and each color is used to generate pixels corresponding to the comparison of signal states with an appropriate threshold distance. The result is a colored recurrence plot, that represents multiple RPs with different threshold distances generated from the same data set.

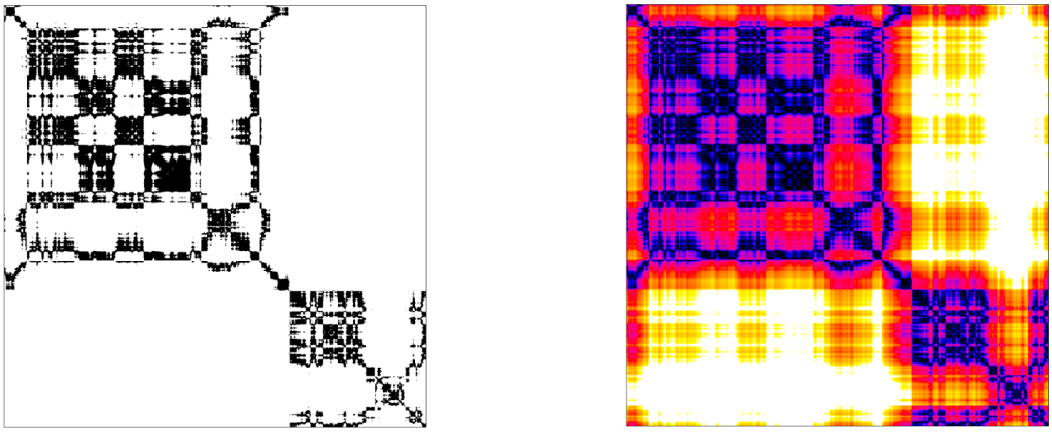


Figure 4. Colorless (on the right) and colored (on the left) recurrence plots generated using Cisco stock time series. Data source: Alpha Vantage [10].

2.7 Data downsampling

The algorithm used to draw recurrence plots requires data to generate an image. Even small amounts of signal values can be used to generate RPs that reveal significant statistical features of the time series. Nevertheless, if the end goal is to observe and interpret recurrence of an entire time series, the algorithm might need to process an arbitrarily large signal.

This poses a potential problem when considering the performance of the application. As data signal size increases, the application will require more time to generate and output the desired recurrence plot. This can be proved by comparing the computation times of an ever increasing data signal.

Table 1. Average computation time comparison.

Signal size	**Avg. computation time (in ms)**
1007	347
1678	987
2517	1470
5034	5332

Table 1 depicts the amount of time on average it takes to draw a recurrence plot using differently sized data signals. Each data signal was run 50 times. As evident by the comparison, the computation times increase exponentially as bigger sized data signals are used. This is due to the implementation of the algorithm within the application. The program takes n^2 times to compare all of the constructed signal states to each other, which translates into proportionally longer computation when generating plots.

However, these developments do not mean that the computational times of generating RPs will remain unpredictable and uncontrollably large as the application is used. While it is true, that the data to be used when generating plots should include as wide a range of values as possible, this does not mean that *all* of the values must be considered for the images to be usable.

In practice, if we consider a narrower sample of a given time series and compare it to the original, the statistical properties and overall rendering of the two data sets remain very similar. This can be observed by inspecting line graphs of the two time series.

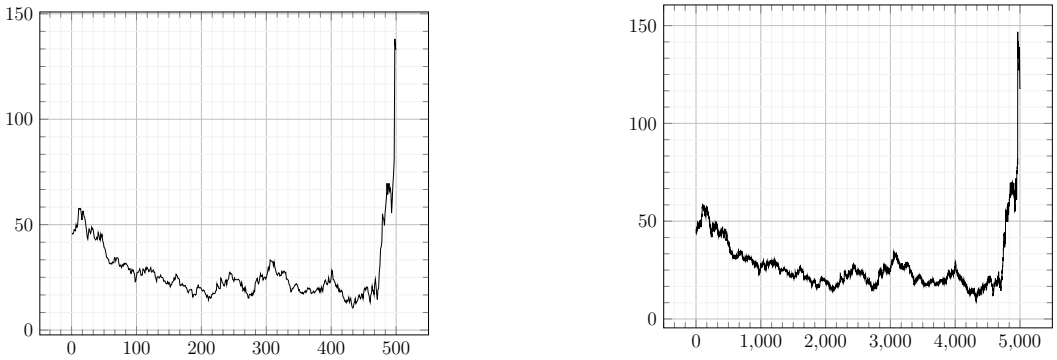


Figure 5. Comparison of CSCO stock price time series of the same range with different data samples (N=500 on the left and N=5000 on the right. Data source: Alpha Vantage [10].

Figure 5 the line graphs of two samples of the same data set. The chart of the left has has been downsampled by including only every 10th value of the original time series (on the right) and thus is ten times smaller. However, as apparent by the the charts themselves, even though a much lower amount of data is used, the renderings show very slight variation. The "peaks" and "valleys" of the charts are similarly placed and structure of the line chart is preserved.

This can be further confirmed by observing the recurrence plots of the two different time series.

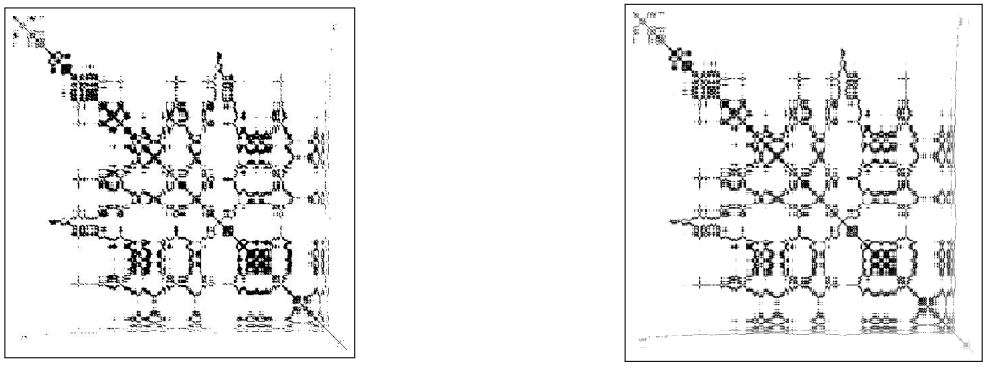


Figure 6. Recurrence plots generated using smaller sample N=500 (on the left) and larger sample N=5000 (on the right). Data source: Alpha Vantage [10].

Figure 6 shows recurrence plots created with the two data sets. The two RPs have very similar appearance, while originating from data sets of different sizes. These different data set sized also lead to the left image being 10 times smaller than right one. Furthermore the smaller image takes a considerably smaller amount of time to be generated (see Table 1).

This kind of data downsampling can be used to control and standardize the amount of time it takes for an RP to be generated. When the user wishes to generate a recurrence plot using any arbitrarily large amount of data, the application can downsample this data by default to drastically decrease computation time. This will of course lead to smaller images being generated, but if the goal is to analyse and interpret the plot, the dimensions of an image are not as important as the structures present.

2.8 Threshold distance calibration

The threshold ε must be adjusted for each recurrence plot. As discussed in section 1.5.3, a threshold can be calibrated by adding and subtracting from the value, until the desired percentage of an RP

is colored. However, manually doing this for each signal processed is a cumbersome task and should not be left as a burden to the end-user. Thus it makes sense to automate this process for each different RP generated.

The implementation of the aforementioned method was achieved by creating a method that was executed each time a recurrence plot was to be generated. The program calculates the percentage of painted area in a recurrence plot generated with a default threshold value ($\varepsilon = 1$) (see Appendice A for code reference). This is then compared to a desired percentage, e.g. 10%. The threshold is increased by 50% if the painted percentage is smaller and decreased by 50% if larger than the desired painted percentage. This processes is repeated until the desired percentage of colored area is reached and the calibrated threshold is used by the program to draw an output RP. An error margin was also applied to the desired painted percentage in order to decrease calculation time.

This method of calibration gives the application flexibility when generating plots, as it focuses on the resulting painted coverage of an RP instead of internals of data. This means, that if desired, the user may adjust the percentage and generate an RP of desired custom structure visibility.

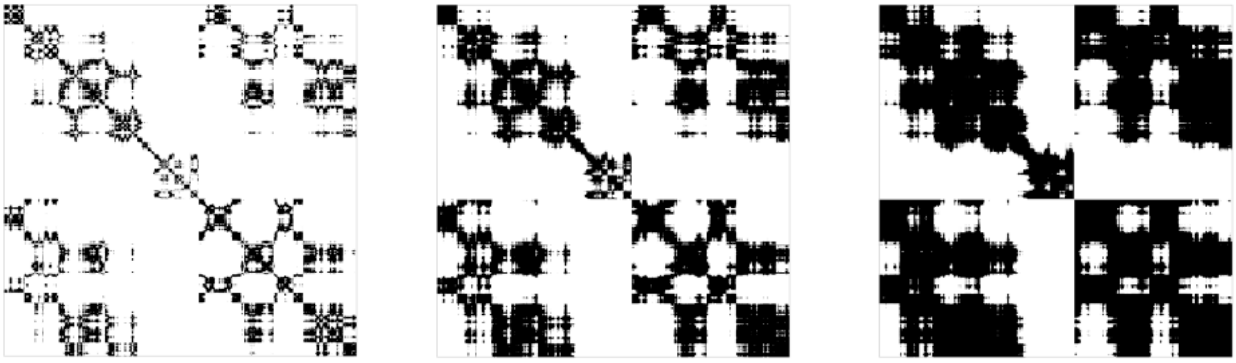


Figure 7. RPs generated using (ADBE) stock time series (N = 1118) with different pixel coverage percentages: 15% (on the left), 30% (centre), 50% (on the left). Data source: Alpha Vantage [10].

Figure 7 depicts the different recurrence plots generated using varying pixel coverage percentages. As seen by the images, an increased percentage leads to recurrence plots having uninformative patterns. Thus the default coverage percentage when generating RPs using the application was left to 15%. This value can be adjusted by the end-user to produce customized plots.

However, it should be noted, that the process of threshold calibration can become lengthy when large amount of data is considered. As signal size grows, so do the dimensions of an RP, making each recurring step of the calibration take longer to complete. The amount of steps taken also increases with larger signals. Luckily, as discussed in section 2.7, the structures of an RP do not change significantly, when smaller samples of data are used to generate the image. Thus, if the sample size is controlled and set to be relatively small, the performance impact can be minimized.

2.9 Recurrence plot view

In order to use stock time series data to generate recurrence plots, a separate specialized view was implemented. In order to navigate to this view, the user must press a "Generate Recurrence Plot" button included in the detailed stock view. When pressed the time series data of the current stock is passed to the recurrence plot view (appendice G and H) and can be used to generate a recurrence plot.

The view contains two sections that separate the creation and analysis functionality of the view. In order to generate a recurrence plot with the desired parameters the user must first select the range of the time series. This can be done by selecting the start and end dates from the two date pickers present in the view. When each of these inputs is selected a number of values present in the data set is presented to the user.

The user may also select the preferred distance calculation norm and type of time series (open, high, low, close) to be used as data. A preferred plot pixel coverage can also be selected by typing the desired percentage in the designated input field. As these options have default values, selecting them is optional. Dimension and time delay embedding parameters can also be applied, however changing these can greatly distort the recurrence plots (as discussed in 1.5.1), thus these options are hidden by default and are accessible by clicking the "Show Advanced Options" button. The user may also choose to create a colored recurrence plot by selecting the designated checkbox.

When all desired options are applied, the recurrence plot is generated by pressing "Generate" button at the bottom the interface. Upon successful generation, the image is displayed to the user. If the colored plot checkbox was selected, a color scale is also presented.

Plots can be interpreted by the user to investigate the statistical properties of a selected time series. For quick reference to observations and their interpretations a button was added, that, upon click, displays them to the user (appendice J).

To perform stationarity analysis to a recurrence plot, a colorless image must first be generated. When this is done, opening the analyse section of the view presents the user with an "Analyse" button, which can be pressed to perform the analysis. When executed, a percentage with relative stationarity percentage of the selected time series is presented along with explanation of its meaning. Due to the implementation of stationarity analysis (see section 3.1), if a recurrence plots cannot be divided into equal amount of squares, the range of selected data is slightly adjusted to generate a plot with dividable dimensions.

3 Studies conducted

3.1 Stationarity analysis

As discussed in section 1.5.5, recurrence plot images can be interpreted by analysing the structures present in the image. The user of the application can interpret RPs by referring to guidelines present in this document and the application.

Recurrence plots can be used reveal statistical features of the times series. The images are used to determine whether or not the time series in question has discernible stationarity. This is quantified to percentage values and presented to the user to provide additional insights about the time series of a stock.

3.1.1 Detecting stationarity

As discussed in section 1.1, stationarity describes whether a change within a system has consistency, i.e. if change happens at a similar rate through out the series.

A time series with high stationarity produces a recurrence plot that has a homogeneous distribution of painted pixels. This means that smaller separated areas of the plot also have roughly the same pixel densities.

This was tested by dividing different recurrence plots into smaller squares and calculating the pixel density of each such square (for code reference see Appendice B.). These resulting values were used to create a histogram of RPs local pixel densities.

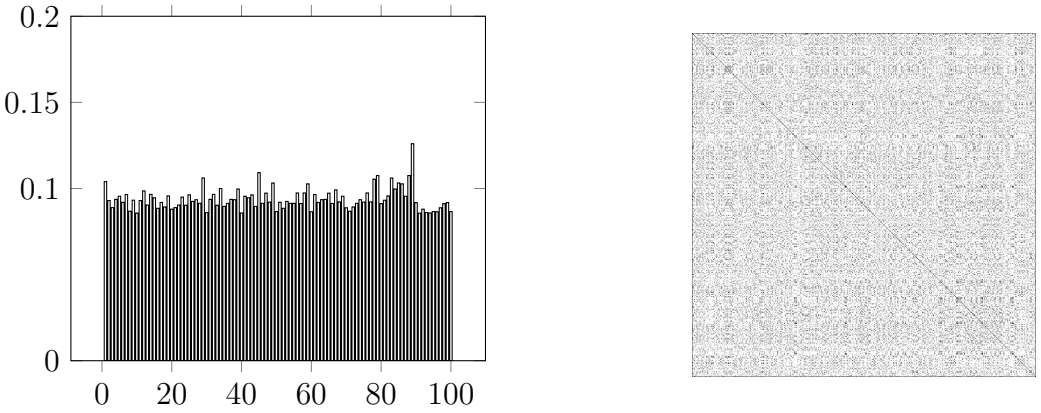


Figure 8. Local pixel density histogram of smaller squares (on the left) of an RP generated from a controlled distribution $x\epsilon(0, 1]$ (on the right). Standard deviation $\sigma = 0.006$.

Figure 8 depicts the histogram of local pixel densities and a homogeneous recurrence plot that produced them. As evident by the distribution of values in the histogram, all square have roughly the same density. This distribution can be measured by calculating the standard deviation σ of these values. The result represents a value which corresponds to the stationarity of the system in terms of local image density.

Compared to a stationary time series, a non stationary (or trending) time series will not have such a homogeneous distribution of local densities.

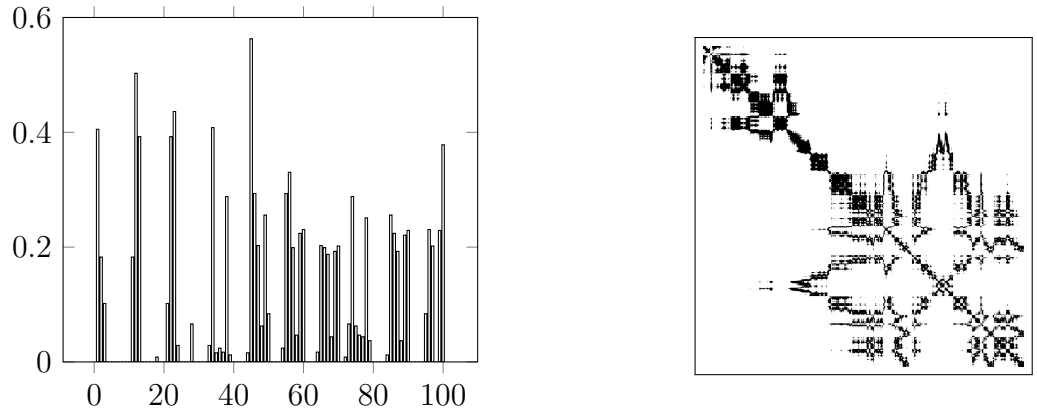


Figure 9. Local pixel density histogram (on the left) of RP generated using Facebook stock time series (on the right). Standard deviation $\sigma = 0.138$.

Figure 9 depicts a RP generated using non-stationary data and its local pixel density histogram. The histogram does not have the same uniformity as seen in figure 8, as was expected. The standard deviation σ value is also higher than that of a stationary system.

If the σ values of two systems, with very high and very low stationarity are determined, these can be used to calculate the stationarity of a desired system as a relative percentage.

The process of stationarity percentage calculation is a straightforward one. When a recurrence plot is generated, the previously used method of stationarity investigation is applied: the plot is divided into smaller squares and their local densities are calculated and used to make a histogram.

Then the standard deviation of the system is calculated which reveals the relative stationarity of the system.

This stationarity can then be compared to high and low stationary σ values by using proportion. If we consider the lowest possible σ to be 0% stationarity and the highest - 100%, then the investigated plot will fall somewhere in between these values.

For instance, the aforementioned Facebook stock time series was investigated and its histogram used to produce a standard deviation $\sigma = 0.138$. This density, when compared between to the maximum possible stationarity time series of uniformly random values present in 8 (which has a standard deviation $\sigma = 0.301$ and the minimum possible stationarity of plot discussed in the upcoming section 3.1.2 (which is a standard deviation $\sigma = 0.227$), comes to roughly 40%.

This value corresponds to the relative stationarity of Facebook stock price fluctuations and reveals how uniform the change is within the time series. A high stationarity suggests that the time series has less volatility when it comes to price changes over time. A low stationarity means that a time series has more volatility and is subject to a higher margin of changes.

The stationarity value of a desired recurrence plot is displayed to the user when a stationarity analysis button is pressed in the recurrence plot window (see section 1.5).

3.1.2 Calibrating minimum stationarity

To start comparing standard deviations of local densities in RPs, maximum and minimum stationarity values must be clearly defined.

The maximum stationarity value is easy to achieve, as the the required pixel density must be uniform trough out the histogram. This can be done by generating and analysing a uniformly stationary time series (such as figure 8).

The difficulty comes when calculating the minimum standard deviation (which represents 0% stationarity). The lack of stationarity will be apparent by a large standard deviation. However, as the signal can be arbitrarily non stationary, it is difficult to determine how large the standard deviation of such a signal might be.

However, as the project focuses on recurrence plots generated with stock time series, minimum stationarity can be determined by examining a large range of such RPs. The lowest highest standard deviation of an analysed plot in this selection can be considered to be the lowest relative stock stationarity.

To achieve this calibration, a large selection of stock symbols need to be examined. Eodata [2], a free global source of stock data was used to collect the stock symbols. The list included stocks from NASDAQ stock exchange, as it is one of the stock exchanges used in this project (see section 1.2).

A total of 3518 stock symbols were used to create a database of recurrence plots using their daily time series. The plots were analysed to calculate standard deviations of their local densities (as discussed in section 3.1). Each plot was downsampled to include around 1000 values so the standard deviations would not be affected by differing data sizes.

The highest relative standard deviation was calculated to be $\sigma = 0.301$. This numbers was used as the base line when calculating the relative stationarity of a stock recurrence plot.

3.2 Distance norm comparison

Distance of recurrence plots can be calculated using three different norms: Euclidean, Manhattan and maximum (see 1.5.4). An implementation and comparison of these norms was made to discover the fundamental differences of their practical application.

The Euclidean norm, which, according to the literature [19] is the most commonly used norm, was implemented first. The calculation procedure depended on the number of dimensions a vector had. At the start of the calculation, this number was extracted from the vector and used to loop over a calculation that many times. The calculation itself made use of *java.lang.Math* class and as the loop was iterating, added squares of vector values together to a local result variable. The method returned a square root of the local result variable.

The second, Manhattan norm, was implemented in a similar way. The iteration also depended on the dimensions of a vector. The difference here was that, as each iteration passed, an absolute value representing the difference of two vectors was added to the end result. This result was the return value of the method.

The third, maximum norm, was implemented similarly as well. The main difference here was that as the loop iterated, an absolute value representing vector difference, was appended to a list. After the loop had completed, the *java.util.Collections* class was used to calculate the maximum value of the list and return it.

The main focus was to compare the execution time of plotting a bitmap and whether the bitmaps themselves showed different structures depending on the norm used.

To investigate the execution time, variables were added before and after a successful plot. These times were returned at the end of an execution and used to calculate the median and average execution time of each norm. The program was executed on a laptop computer described in section 2.4.

Table 2. Results of generating recurrence plots using different data sets and distance calculation methods. RP $\varepsilon = 1.5$ and the number of executions was $N = 500$.

Signal type	**Data range**	**Norm**	**Average time (in ms)**	**Median**
Sin(x) function	720	Euclidean	315	316
		Manhattan	213	312
		Maximum	208	234
Sin(x) function	2160	Euclidean	2115	3163
		Manhattan	1958	2918
		Maximum	1868	2793
MSFT stock daily prices	3000	Euclidean	4356	6459
		Manhattan	3757	5488
		Maximum	3674	5388

Table 2 shows the results of the experiment. The Maximum and Manhattan norms displayed the shortest generation times. The Euclidean norm was consistently the worst performing norm to be used in both average running times and medians.

Figure 10 shows a comparison between recurrence plots generated using different norms. A lower range of data was used to highlight the differences between RPs. As evident by the visual appearance, maximum norm contained the highest amount of pixels, while Manhattan norm - the lowest. The patterns depicted in these images did not show significant difference.

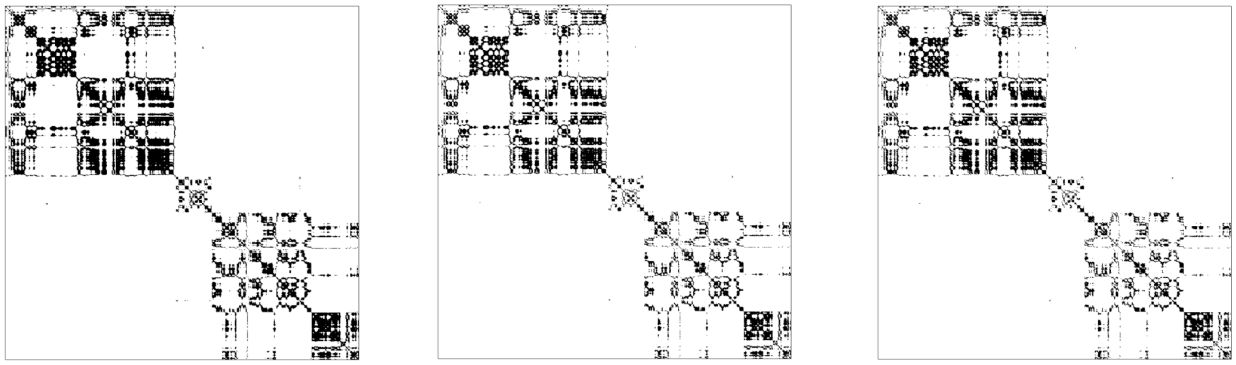


Figure 10. Comparison of GOOGL stock recurrence plot (N=682) drawn with different norms: Euclidean (on the left), Manhattan (centre) and Maximum (on the right).

3.3 High dimensional plots comparison

When generating and analysing recurrence plots a thing to consider is the dimension D and time delay d parameters of the algorithm (see section 1.5.1). It is worth investigating whether changing these parameters impact the resulting recurrence plots in a significant way.

To test whether notable notable appear in generated images, recurrence plots using ever increasing time delay and dimension parameters were created using time series of different stocks.

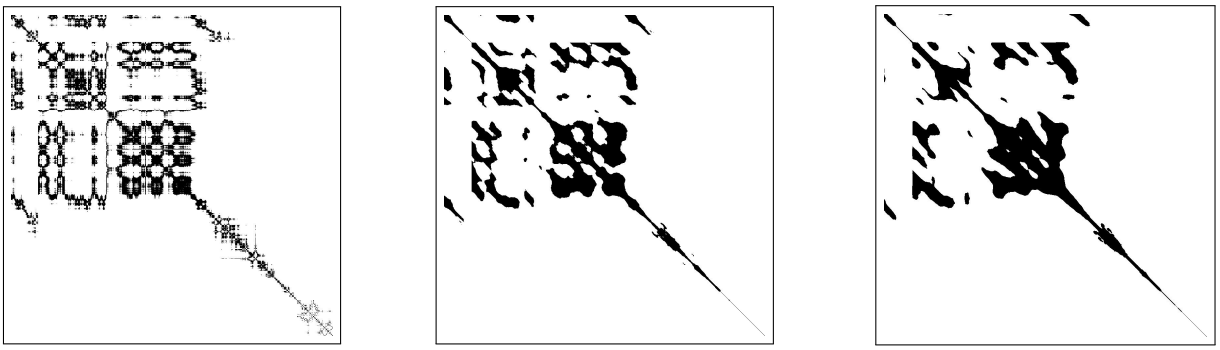


Figure 11. Recurrence plots generated using Adobe (ADBE) stock time series (N = 1000) with parameters $D = 1$, $D = 25$ and $D = 50$ respectively. Data source: Alpha Vantage [10].

Figure 11 depicts how differently an RP is drawn when using an ever increasing dimension and a static time delay parameters. In theory, as dimension parameter D grows larger, the differences between resulting signal states become larger and thus require a larger threshold parameter ε for them to be considered recurrent. However, as the the threshold is calibrated automatically, this constant growth is not noticeable in practice. That is why the resulting plots, albeit differing in texture, portray roughly similar structures.

It is also worth mentioning that, as dimension variable increases, the dimensions of the resulting image decreases, because less signal states are constructed and compared when generating the plot.

Figure 12 shows a comparison of RPs drawn using signal states with ever increasing time delay and dimension parameters. As seen by the comparison, the plots become greatly distorted as the parameters increase. Excessively high parameters make the plot indiscernible and should be avoided. It should also be noted that dimensions of the plot shrank dramatically faster when both parameters were increased at the same time.

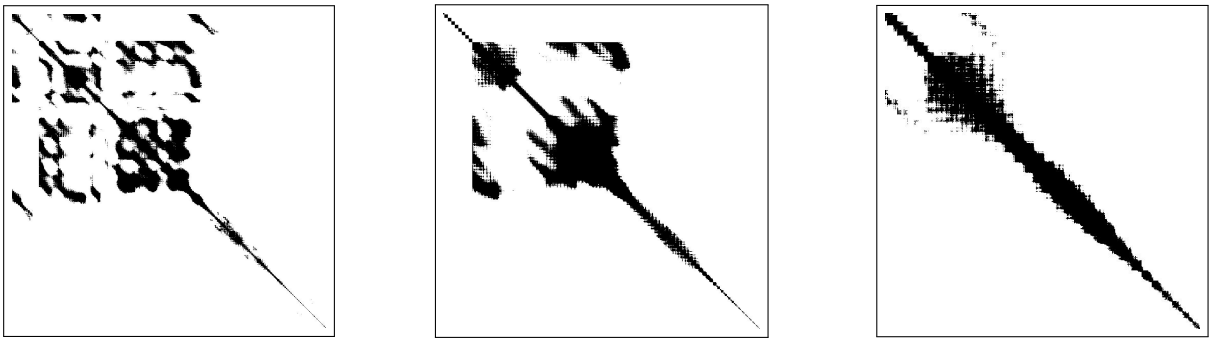


Figure 12. Recurrence plots generated using Adobe (ADBE) stock time series (N = 1000) with parameters $D, d = 10$, $D, d = 20$ and $D, d = 30$ respectively. Data source: Alpha Vantage [10].

These comparisons reveal that applying increasing values of parameters D and d increasingly distort the structures in recurrence plots. According to the literature [19], the parameters $D = 2$ and $d = 1$ are most commonly used to draw RPs. This will be used as the default option when plots are generated. However as the user might want to generate a plot with different parameters, this will be left as an option in the application as well.

Conclusions and Recommendations

During the course of this project, recurrence plots and their generation implementation was examined in detail. In the final application, all proposed functionality was implemented including common stock observation and categorisation features (discussed in section 2.5), recurrence plot generation (see section 2.6) and stationarity analysis (see section 3.1). The following is the summary and results of investigations performed in this project:

- Recurrence plots were analysed to calculate the local pixel densities of smaller squares within the image. The standard deviation of these densities was calculated and compared with those of other recurrence plots. This approach proved to be suitable for calculating the relative stationarity of the time series used to generate the plot (see section 3.1.1).
- When generating recurrence plots, Euclidean, Maximum and Manhattan norms were used to calculate whether time series signal states were within a selected threshold distance. Signals of different lengths were used to compare the performance impact of each norm (see 3.2). It was concluded that the Maximum and Manhattan norms had the shortest completion times when generating plots. No substantial visual differences were observed when comparing the generated images.
- Recurrence plots were be generated with varying time delay and dimension variable values. Relatively high dimension and time delay variable values led to recurrence plots being distorted and uninformative, eliminating the possibility of using them to determine statistical properties of the signal. When only the dimension variable was increased, recurrence plots showed less significant differences. The default variable values used in the application were chosen as proposed in the analysed literature (read more on section 11).

The functionality of the application can still be improved. The addition of Recurrence Quantification Analysis can provide additional signal information when analysing recurrence plots. Current implementation analyses recurrence plots by manually calculating image pixel densities. This could be improved by applying Machine Learning algorithms to set up automatic recurrence plot structure detection.

References

[1] Android studio.
https://developer.android.com/studio/.

[2] Eodata. stock symbol list.
http://eoddata.com/symbols.aspx.

[3] Exchange rates api.
https://exchangeratesapi.io/.

[4] Fasterxml jackson core.
https://github.com/FasterXML/jackson.

[5] Gsm arena. huawei p30.
https://www.gsmarena.com/huawei_p30-9530.php.

[6] Loopj android async http.
https://github.com/android-async-http/android-async-http.

[7] Material search view.
https://github.com/MiguelCatalan/MaterialSearchView.

[8] Mp android chart.
https://github.com/PhilJay/MPAndroidChart.

[9] Recurrence plots and cross recurrence plots.
http://www.recurrence-plot.tk/.

[10] Stock quotes and historical data.
https://www.alphavantage.co/.

[11] *Material Design*.
https://material.io/design/.

[12] *What Causes Stock Prices to Change?* Desjardins Online Brokerage.
https://www.disnat.com/en/learning/trading-basics/stock-basics/
what-causes-stock-prices-to-change.

[13] Web archive. model view controller.
https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.
html.

[14] *About Stock Charts*. Tidestone Technologies Inc, 1999.
https://www.mit.edu/~mbarker/formula1/f1help/11-ch-10.htm.

[15] *Common Stock vs. Preferred Stock, and Stock Classes*. Investors Guide, 2019.
http://www.investorguide.com/article/11166/stock-classes-the-difference-between-common-stock-ar

[16] *Stock Exchange Rankings*. I Know First. Daily Market Forecast, 2019.
https://iknowfirst.com/ranking-the-top-10-worldwide-stock-exchanges-us-exchanges-maintain-stron

[17] Paul Dawkins. *Heaviside function*, 1999. http://tutorial.math.lamar.edu/Classes/DE/StepFunctions.aspxm.

[18] Vytautas Kondratas. *Stock market price observation system*. Faculty of Mathematics and Informatics, Vilnius University, 2019.

[19] Tadas Meškauskas. *Signalu analize ir apdorojimas*. Vilniaus Universiteras. Matematikos ir Informatikos fakultetas, 2017. https://klevas.mif.vu.lt/~meska/SAA/Tadas_Meskauskas_-_Signalu_Analize_Ir_Apdorojimas_-_Mokymo_Priemone.pdf.

[20] Marco Thiel Jürgen Kurths Norbert Marwan, M. Carmen Romano. *Recurrence plots for the analysis of complex systems*. Institute of Physics, University of Potsdam, January 2007.

[21] Shay Palachy. *Stationarity in time series analysis*. Towards Data Science, April 8 2002. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322.

[22] Don Percival. *Introduction to Spectral Analysis*. Applied Physics Lab, University of Washington.

[23] Richard A. Davis Peter J. Brockwell. *Introduction to Time Series and Forecasting*. Springer, 2002.

[24] Xiao Gang Su Xin Yan. *Linear Regression Analysis. Theory and Computing*. World Scientific, 2009.

Appendices

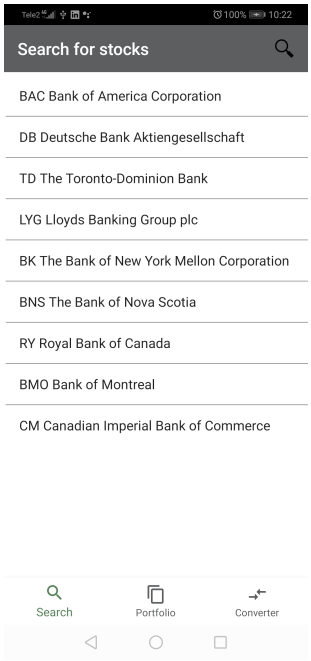
A. Painted area calculation method

```
1 private double calculatePaintedPercentage(double threshold) {
2     int paintedPixels = 0;
3     for (int i = 0; i < signalStates.size(); i++) {
4         for (SignalState signalState : signalStates) {
5             if (isSignalRecurrent(signalStates.get(i), signalState, this
                  .distanceType, threshold)) {
6                 paintedPixels++;
7             }
8         }
9     }
10    int totalPixels = imageSize * imageSize;
11    return (double) paintedPixels / totalPixels;}}
```

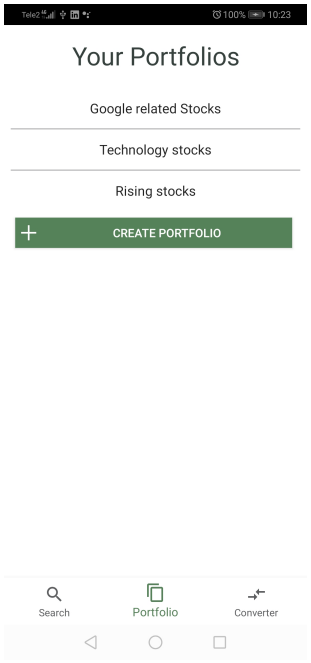
B. Image division method

```
1 private List<BufferedImage> divideImage(BufferedImage image) {
2     List<BufferedImage> subImages = new ArrayList<>();
3     int height = image.getHeight();
4     if (640000 % height != 0)
5         throw new IllegalArgumentException("Image cannot be divided into
              equal squares");
6
7     for (int j = 0; j < image.getWidth(); j += SUB_SQUARE_DIMENSION) {
8         for (int i = 0; i < image.getHeight(); i += SUB_SQUARE_DIMENSION
              ) {
9             subImages.add(image.getSubimage(j, i, SUB_SQUARE_DIMENSION,
                  SUB_SQUARE_DIMENSION));
10        }
11    }
12    System.out.println("Divided into " + subImages.size() + " images");
13    return subImages;}
```

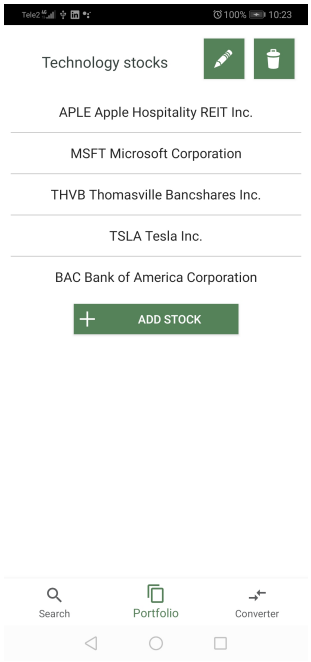
C. Search view



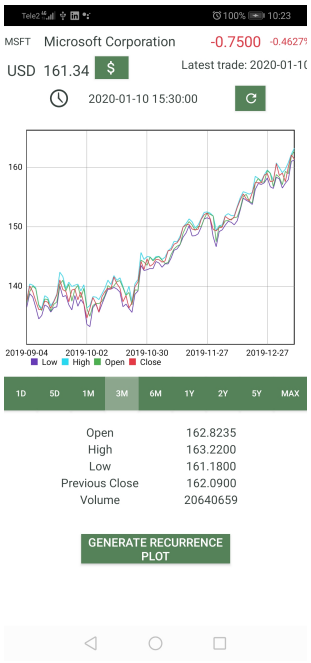
D. Portfolio list view



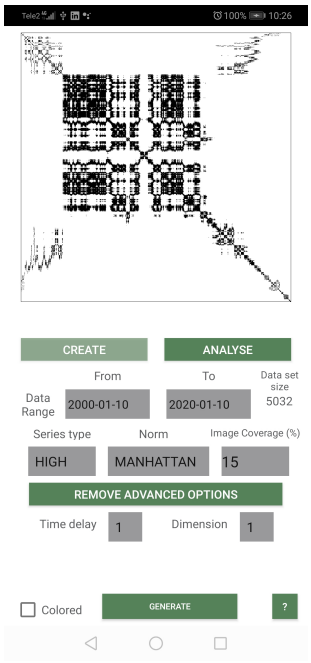
E. Portfolio Item view



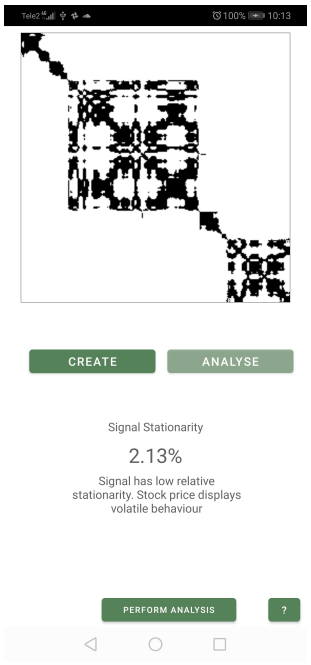
F. Stock view



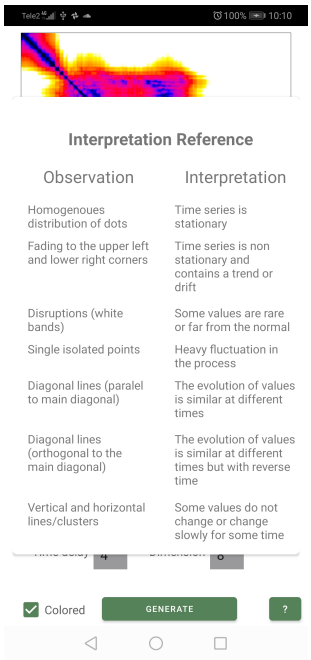
G. Recurrence plot creation section



H. Recurrence plot analysis section



J. Interpretation pop up



K. Currency converter view