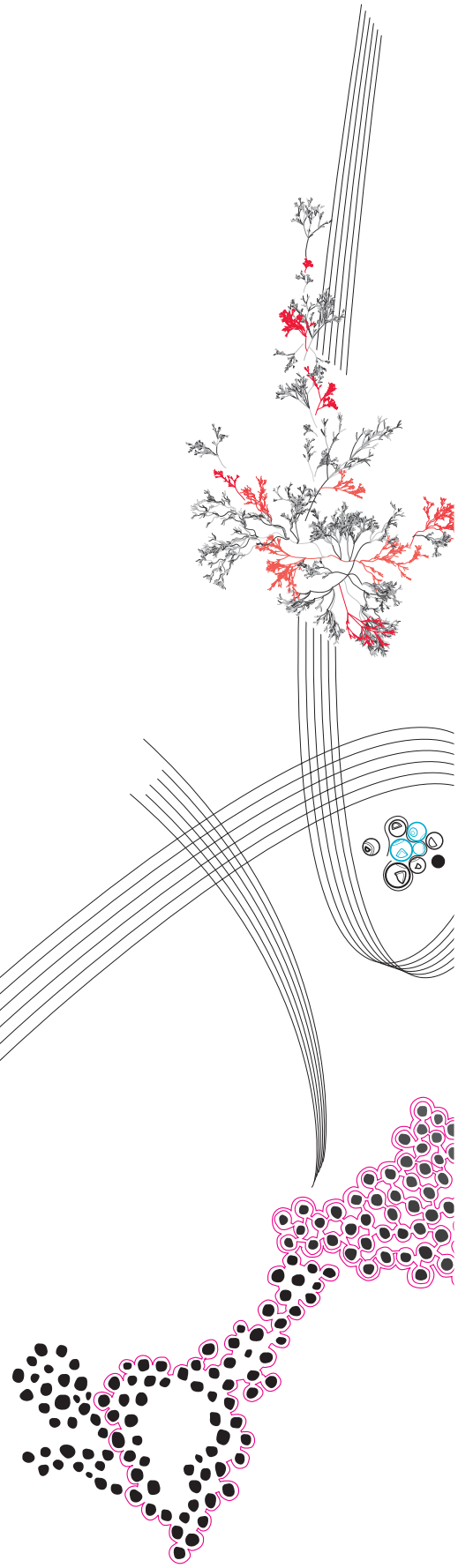

BSc Thesis Applied Mathematics

Comparing ordinal pattern-based short-term predictions to alternative forecasting methods

Herkus Jacina

Supervisors: Annika Betken and Giorgio Micali

July, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I would like to express my sincere gratitude to my supervisors for introducing me to the intriguing and fascinating mathematical discipline known as time series analysis and providing guidance throughout the entire research process. This shared knowledge allowed me to understand and learn more about how, in this data-driven world, the data can be analysed and processed to draw conclusions and make predictions.

Comparing ordinal pattern-based short-term predictions to alternative forecasting methods

Herkus Jacina

July, 2024

Abstract

In this work, the feasibility of using ordinal patterns for short-term prediction is analysed. The accuracy of ordinal pattern-based predictions is compared to the accuracy of the linear predictor method. These comparisons are then used to conclude under which assumptions and conditions the predictions made by the two methods are similar. The considered stochastic processes are: moving averages, autoregressive, and fractional Brownian motion. Additionally, real-world data sets are used to analyze how the prediction methods perform.

Keywords: Ordinal patterns, Predictions, Linear predictors, Fractional Brownian motion, Fractional Gaussian noise, Hurst parameter, Moving average processes, Autoregressive processes, Real-world data

1 Introduction

Ordinal patterns are sequences of symbols that represent the relative order of elements within a time series or dataset, rather than their exact values. In earlier papers, ordinal patterns were referred to as permutations and were used to analyse the complexity or predictability of time series using a measurement called permutation entropy [1]. As more research surfaced on ordinal patterns, it became a useful mathematical concept in fields that analyse time series. In medicine, ordinal patterns are used to analyse and draw conclusions for EEG data [2], heartbeat [3], or patients with severe injuries [4]. This showcases the versatility of ordinal patterns and the benefit of using patterns to analyse time series.

In recent times, ordinal patterns have started to be used as a predictor for time series. In the article "Short-term prediction through ordinal patterns" by Yair Neuman, Yochai Cohen and Boaz Tamir [5], it is analysed whether ordinal patterns can be used as a time series forecasting method. While the work focused on predicting the next pattern in the time series, the results showcased that the accuracy of predictions made were better than randomly guessing the next ordinal pattern. This provided a basis to consider ordinal patterns as a prediction method and since then other papers have been published analysing the feasibility of using ordinal patterns for short-term predictions.

In this paper, research is conducted to analyse whether ordinal patterns can be used as a short-term predictor. Additionally, the predictions made using ordinal patterns and their accuracy are compared to predictions made using the linear predictions method. This comparison aims to identify the conditions and assumptions under which ordinal pattern predictions are as accurate as predictions made by alternative forecasting methods.

2 Preliminaries

The research topic involves concepts from many different disciplines of mathematics. Therefore, it is important to define them well, to ensure that readers from any background can follow the derivations and results of this research. This section introduces and briefly discusses the concepts relevant to this research.

2.1 Time series and Stochastic processes

Since the research focuses on analysing multiple prediction methods for time series, it is important to understand what time series are and the processes that generate these time series.

A time series is a sequence of measurements recorded at discrete time instants $t \in T$, where T is a discrete set, and is usually denoted $(X_t)_{t\in T}$ [6]. Data points in a time series are observations of a single random variable or group of variables across time, allowing for the examination of the dynamic and establishing trends between the observations. Formally, a time series is defined as [14]:

Definition 2.1 (Time Series) A time series is a stochastic process $(X_t)_{t\in T}$, where T is a set of times. We call $(X_t)_{t\in T}$ a realization of the process.

Time series analysis refers to methods for analyzing time series data in order to derive useful statistics and data properties, which can be used for making time-series predictions.

We see a time series as a realizations $(x_t)_{t\in T}$ of a stochastic process. These random variables represent time series evolution over a set period of time, with the unpredictable nature of the future observations. Stochastic process is formally defined as [14]:

Definition 2.2 (Stochastic process) Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and (E, ϵ) a measurable space, called *state space*. Let T be an index set. A stochastic process $X = (X_t)_{t\in T}$ defined on $(\Omega, \mathcal{F}, \mathbb{P})$ with values in (E, ϵ) is a function:

$$X : \Omega \times T \to E \text{ s.t, } \forall t \ \ \omega \to X_t(\omega) \text{ is measurable from } (\Omega, \mathcal{F}) \text{ to } (E, \epsilon).$$

Stochastic processes are commonly used in scientific disciplines such as physics, economics, and engineering to represent systems that vary over time as a result of random events.

2.2 (Sample) Mean, Variance, and Autocovariance

Mean value and variance are powerful tools not only to analyse the results and draw conclusions but also to characterize the distribution of time series [7].

Mean value is a statistical measurement that represents the average value in a set of observations and we denote it as $\mathbb{E}(X_t)$. For time series that implies finding the average of observed values $\{x_1, x_2, ..., x_T\}$ over a time period T [7]. The resulting average of time series observations is called a sample mean and is denoted using notation $\hat{\mu}$.

Definition 2.3 (Sample mean) Let X be time series indexed by T. Then, the sample mean $\hat{\mu}$ for the time series X is the random variable:

$$\hat{\mu}(X) = \frac{1}{|T|} \sum_{t=1}^{|T|} x_t.$$

Variance is a measure that describes the spread in a set of observations around its mean [7]. Variance can be denoted as $\text{var}(X_t)$ or σ^2, where we call σ the standard deviation. The resulting spread of time series observations is called a sample variance and is denoted using notation $\hat{\sigma}^2$.

Definition 2.4 (Sample variance) Let X be time series indexed by T. Then, the sample variance $\hat{\sigma}^2$ for the time series X is calculated using the following expression:

$$\hat{\sigma}^2(X) = \frac{1}{|T|} \sum_{t=1}^{|T|} (X_t - \mu)^2.$$

Lastly, autocovariance measures the joint variability of two variables at distance k, which we refer to as lag k. We denote autocovariance at lag k as $\gamma(k)$. The resulting joint variability of time series observations at lag k is called a sample autocovariance at lag k and is denoted using notation $\hat{\gamma}(k)$.

Definition 2.5 (Sample autocovariance) Let X be time series for a time period T. Then, the sample autocovariance at lag k, $\hat{\gamma}(k)$, for the time series X is calculated using the following expression:

$$\hat{\gamma}(k) = \frac{1}{T-k} \sum_{t=1}^{T-k} (X_t - \mu)(X_{t+k} - \mu).$$

Furthermore, we introduce the following definition for stochastic processes X_t [14]:

Definition 2.6 (Wide-sense stationary) We refer to stochastic processes as wide-sense stationary (WSS), if they satisfy the following conditions:

- $\mathbb{E}(X_t) = \mathbb{E}(X_1)$ for all t;
- $\mathbb{E}(X_t^2) < +\infty$;
- $Cov(X_t, X_s) = f(t-s)$ for some real valued function f.

The mean value function being constant implies that the $\mathbb{E}(X_t)$ is independent of the time variable t in a time series. Additionally, the wide-sense stationary process has a finite variance. Lastly, the autocovariance being time-invariant means that the value depends only on the time difference $t - s$ and not on the specific time t and s.

The WSS condition is important when considering time-series predictions. This ensures that predictions have scientific reasoning and are not just random predictions. Therefore checking whether the stochastic process is WSS is crucial, before using time-series prediction methods.

A special WSS process is the white noise, defined as [14]:

Definition 2.7 (White noise) A stochastic process ϵ_t, $t \in T$, is white noise if $\mathbb{E}(\epsilon_s) = \mathbb{E}(\epsilon_t)$ for all $s, t \in T$, $\gamma_\epsilon(0) = \sigma_\epsilon^2 \geq 0$ and $\gamma_\epsilon(\tau) = 0$ for all $\tau \neq 0$.

White noise refers to a process of uncorrelated random variables that follow a fixed distribution with a constant mean μ and variance σ^2 [7]. This implies that the autocovariance function $\gamma(k) = 0$ for $k \neq 0$ and $\gamma(0) = \sigma^2$. Typically we use notation ϵ to denote white noise and say that standard white noise values follow a $N(0, 1)$ distribution, meaning that the mean value is assumed to be 0 and standard deviation σ to be equal to 1.

2.3 Fractional Brownian motion and Fractional Gaussian noise

In this thesis, we will make use of fractional Brownian motion (fBm) and fractional Gaussian noise (fGn) to simulate artificial time series on which we will compare the time series prediction methods.

The fBm process is a continuous-time Gaussian process with a fraction parameter $H \in (0, 1]$ which is referred to as the Hurst parameter [8]. The increment between two fBm values X_{t-1} and X_t is called fractional Gaussian noise, which we denote as Y_t and satisfies the following expression $Y_t = X_t - X_{t-1}$, where X_t are fBm. The fGn is normally distributed with $\mathbb{E}(Y_t) = 0$ for all $t \in \mathbb{N}$ and autocovariance which satisfies the following equation:

$$\gamma(k) = \frac{1}{2}\left(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}\right). \tag{1}$$

2.4 Ordinal patterns

As mentioned in the introduction, ordinal patterns refer to an arrangement of time series observations in a partition based on their relative rank. The order of ordinal patterns refers to the number of observations in a partition. We consider the use of ordinal patterns of order 3, meaning the relative rank of three consecutive observations (X_t, X_{t+1}, X_{t+2}). For instance for order 3 the following are all ordinal patterns and the conditions they satisfy:

$$\begin{aligned}
\pi_1 = (0,1,2) &\quad \Leftrightarrow \quad X_t < X_{t+1} < X_{t+2};\\
\pi_2 = (0,2,1) &\quad \Leftrightarrow \quad X_t < X_{t+1} \;\&\; X_{t+1} > X_{t+2} \;\&\; X_t < X_{t+2};\\
\pi_3 = (1,0,2) &\quad \Leftrightarrow \quad X_t > X_{t+1} \;\&\; X_{t+1} < X_{t+2} \;\&\; X_t < X_{t+2};\\
\pi_4 = (1,2,0) &\quad \Leftrightarrow \quad X_t < X_{t+1} \;\&\; X_{t+1} > X_{t+2} \;\&\; X_t < X_{t+2};\\
\pi_5 = (2,0,1) &\quad \Leftrightarrow \quad X_t > X_{t+1} \;\&\; X_{t+1} < X_{t+2} \;\&\; X_t > X_{t+2};\\
\pi_6 = (2,1,0) &\quad \Leftrightarrow \quad X_t > X_{t+1} > X_{t+2}.
\end{aligned}$$

3 Linear predictors for fractional Gaussian noise

This section explains how to derive the best linear predictor, with a focus on fractional Gaussian noise. We then develop these predictors and evaluate their accuracy using metrics such as minimum mean squared error and mean squared error. These measures aim to demonstrate the accuracy of linear prediction methods when dealing with fractional Gaussian noise.

3.1 Derivation of best linear predictor

A linear predictor is one way to predict the next value in the time series. A linear predictor is a weighted sum of the past observations.

Let $\hat{X}_{t,n}$ denote a linear predictor of the time series at time t, using n previous observations, i.e. a random variable of the form:

$$\hat{X}_{t,n} := \sum_{i=1}^{n} a_i X_{t-i},$$

where a_i denotes the coefficients of this weighted sum.

Thus the objective becomes determining the coefficient a_i that minimize the mean squared error (MSE): $\mathbb{E}(X_t - \hat{X}_{t,n})^2$. The predictor $\hat{X}_{t,n}$ that minimizes MSE is called the best linear predictor [14].

Definition 3.1 The $\hat{X}_{t,n}$ for which MSE is minimized, is called the best linear predictor.

To determine the coefficients a_i that minimize the MSE, we let a_0 = -1, and then MSE can be rewritten as [14]:

$$\mathbb{E}\left(\sum_{i=0}^{n} a_i X_{t-i}\right)^2. \tag{2}$$

Under assumption that $\mathbb{E}(X_t) = 0$ for all t, the equation (2) can be expressed as:

$$\mathbb{E}\left(\sum_{i=0}^{n} a_i X_{t-i}\right)^2 = \sum_{i=0}^{n}\sum_{j=0}^{n} a_i a_j \gamma(i-j),$$

where γ denotes the autocovariance function. Because the objective is to minimize this quadratic equation with respect to the coefficients $a_1, a_2, ..., a_n$, partial differentiation with respect to a_k, $k = 1, 2, ..., n$ yields the desired conditions for minimality [14]:

$$\sum_{i=0}^{n} a_i \gamma(i-k) = 0, \quad k = 1, 2, ..., n. \tag{3}$$

Because we let a_0 = -1, the expression (3) can be rewritten as:

$$\gamma(k) = \sum_{i=1}^{n} a_i \gamma(i-k), \quad k = 1, 2, ..., n.$$

This set of equations in matrix form becomes

$$\begin{bmatrix} \gamma(0) & \gamma(1) & \gamma(2) & \dots & \gamma(n-1) \\ \gamma(1) & \gamma(0) & \gamma(1) & \ddots & \vdots \\ \gamma(2) & \gamma(1) & \gamma(0) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \gamma(n-1) & \dots & \dots & \dots & \gamma(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \gamma(3) \\ \vdots \\ \gamma(n) \end{bmatrix}. \tag{4}$$

We compactly rewrite (4) as $\Gamma_n a = \gamma_n$. These equations are called the Yule-walker equations. It can be observed, that the autocovariance matrix Γ_n on the left side of the expression (4) is symmetric. Because the resulting matrix is symmetric, the following theorem can be used to conclude, that the matrix is diagonalizable and admits real eigenvalues [15].

Theorem 3.1 (Spectral theorem) In a finite-dimensional Euclidean space, every symmetric transformation has an orthonormal eigenbasis.

Under the assumption that the resulting matrix is positive semi-definite, the conditions for matrix inversion can be formulated as the following theorem [11]:

Theorem 3.2 If matrix A is positive semi-definite, then matrix A is invertible if-and-only-if all the eigenvalues are positive.

Therefore to be able to conclude that the autocovariance matrix Γ_n is invertible, we need to derive the restrictions placed on values of the matrix that need to be satisfied. To derive these restrictions, Gershgorin's theorem is used [12]:

Theorem 3.3 (Gershgorin's theorem) Every eigenvalue of a square matrix A satisfies:

$$|\lambda - A_{ii}| \leq \sum_{j \neq i} |A_{ij}| \quad i \in \{1, 2, ..., n\}. \tag{5}$$

Since for all $i \in 1, 2, ..., n-1$, the values A_{ii} in the autocovariance matrix Γ_n are equal to $\gamma(0)$, equations (5) can be rewritten as:

$$|\lambda - \gamma(0)| \leq \sum_{j \neq i} |A_{ij}| \quad i \in \{1, 2, ..., n-1\}.$$

Thus, we get that the value $\lambda - \gamma(0)$ is bounded, thus giving the following inequality:

$$-\sum_{j \neq i} A_{ij} \leq \lambda - \gamma(0) \leq \sum_{j \neq i} A_{ij}.$$

Adding $\gamma(0)$ to the equation results in the following inequality:

$$-\sum_{j \neq i} A_{ij} + \gamma(0) \leq \lambda \leq \sum_{j \neq i} A_{ij} + \gamma(0). \tag{6}$$

For the matrix to be invertible, none of the eigenvalues should be zero. To ensure this, the interval for λ should not include zero, which implies:

$$\gamma(0) - \sum_{j \neq i} A_{ij} > 0.$$

Which can be rewritten as follows:

$$\gamma(0) > \sum_{j \neq i} A_{ij} \geq \sum_{i=1}^{n-1} \gamma(i),$$

where the last inequality follows from observing the first row of the autocovariance matrix Γ_n. Therefore, we derive the following theorem:

Theorem 3.4 Autocovariance matrix Γ_n of Yule-walker equations (4) is invertible if

$$\gamma(0) > \sum_{i=1}^{n-1} \gamma(i).$$

If the covariance matrix Γ_n is invertible, then multiplying both sides of the Yule-Walker equations (4) by the inverse of this matrix gives the set of equations for the coefficients a_i that minimizes the MSE $\mathbb{E}(X_t - \hat{X}_{t,n})^2$ i.e $a = \Gamma_n^{-1}\gamma_n$.

3.2 Best linear predictor for fractional Gaussian noise

Since fractional Gaussian noise equals $Y_t = X_t - X_{t-1}$, where X_t is fractional Brownian motion, we can determine whether the next observed value X_t was higher or lower than X_{t-1} using linear predictor for Y_t. We consider a linear predictor for fractional Gaussian noise using n previous observations:

$$\hat{Y}_{t,n} := \sum_{i=1}^{n} a_i Y_{t-i}.$$

By definition of fGn, $\mathbb{E}(Y_t) = 0$ for all t, therefore we can use Yule-walker equations (4) to determine the coefficients a_i that minimize the MSE $\mathbb{E}(Y_t - \hat{Y}_{t,n})$. Additionally, the expressions for autocovariance for fGn are known and can be calculated if the Hurst parameter H is known. However, calculating the inverse of a square matrix becomes difficult when dimensions exceed the value of 3. Therefore, for the research, we only consider linear predictors for fGn that uses 1, 2, or 3 previous observations.

Thus we get that the best linear predictor that uses 1 previous observation is of the following form:

$$\hat{Y}_{t,1} = a_1 Y_{t-1}.$$

Whereby multiplying the Yule-Walker equations (4) by the inverse of covariance matrix Γ_1, we get the following expression for a_1:

$$a_1 = \gamma(0)^{-1}\gamma(1) = \frac{\gamma(1)}{\gamma(0)} = \gamma(1),$$

where the last equality follows from the fact that for fractional Gaussian noise $\gamma(0) = 1$. Therefore, the best linear predictor that uses 1 previous observation is $\hat{Y}_{t,1} = \gamma(1)Y_{t-1}$.

Theorem 3.5 If Y_t is fractional Gaussian noise, then linear predictor that uses 1 previous observation $\hat{Y}_{t,1}$ is the best linear predictor if-and-only-if $a_1 = \gamma(1)$ for $\hat{Y}_{t,1} = a_1 Y_{t-1}$.

Now consider a linear predictor for fractional Gaussian noise that uses 2 previous observations. The linear predictor is of the following form:

$$\hat{Y}_{t,2} = a_1 Y_{t-1} + a_2 Y_{t-2}.$$

Once again by using Yule-Walker equations (4), we can solve the set of equations to determine the coefficients a_1 and a_2. This is the resulting set of equations:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix}^{-1} \begin{bmatrix} \gamma(1) \\ \gamma(2) \end{bmatrix}.$$

The inverse of a 2x2 matrix on the right side of the equation is calculated as follows:

$$\begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix}^{-1} = \frac{1}{\gamma(0)^2 - \gamma(1)^2} \begin{bmatrix} \gamma(0) & -\gamma(1) \\ -\gamma(1) & \gamma(0) \end{bmatrix} = \begin{bmatrix} \frac{\gamma(0)}{\gamma(0)^2-\gamma(1)^2} & -\frac{\gamma(1)}{\gamma(0)^2-\gamma(1)^2} \\ -\frac{\gamma(1)}{\gamma(0)^2-\gamma(1)^2} & \frac{\gamma(0)}{\gamma(0)^2-\gamma(1)^2} \end{bmatrix}.$$

It should be noted that $\gamma(1)^2 \neq \gamma(0)^2$ for all values of H.

Then multiplying the two matrices on the right side of the equation results in coefficients a_1 and a_2 that constitute the best linear predictor $\hat{Y}_{t,2}$. The coefficients have the following expressions:

$$\begin{aligned} a_1 &= \tfrac{\gamma(1)(\gamma(0)-\gamma(2))}{\gamma(0)^2-\gamma(1)^2}; \\ a_2 &= \tfrac{\gamma(0)\gamma(2)-\gamma(1)^2}{\gamma(0)^2-\gamma(1)^2}. \end{aligned} \tag{7}$$

Theorem 3.6 If Y_t is fractional Gaussian noise, then linear predictor that uses 2 previous observation $\hat{Y}_{t,2}$ is the best linear predictor if-and-only-if a_1 and a_2 are equal to the values in expression (7) for $\hat{Y}_{t,2} = a_1 Y_{t-1} + a_2 Y_{t-2}$.

Lastly, the linear predictor for fractional Gaussian noise that uses 3 previous observations, has the following expression:

$$\hat{Y}_{t,3} = a_1 Y_{t-1} + a_2 Y_{t-2} + a_3 Y_{t-3}.$$

Once again we use Yule-walker equations (4) to determine the coefficients a_1, a_2, and a_3:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) & \gamma(2) \\ \gamma(1) & \gamma(0) & \gamma(1) \\ \gamma(2) & \gamma(1) & \gamma(0) \end{bmatrix}^{-1} \begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \gamma(3) \end{bmatrix}.$$

Finding the inverse of a 3x3 matrix (See appendix A) and matrix multiplying the two matrices on the right side of the equation results in coefficients a_1, a_2, and a_3 that constitute the best linear predictor $\hat{Y}_{t,3}$. The coefficients have the following expressions:

$$\begin{aligned} a_1 &= \tfrac{\gamma(1)(\gamma(0)^2-\gamma(1)^2)+\gamma(1)\gamma(2)(\gamma(2)-\gamma(0))+\gamma(3)(\gamma(1)^2-\gamma(0)\gamma(2))}{\gamma(0)(\gamma(0)^2-\gamma(1)^2)-\gamma(1)^2(\gamma(0)-\gamma(2))+\gamma(2)(\gamma(1)^2-\gamma(0)\gamma(2))}; \\ a_2 &= \tfrac{\gamma(1)^2(\gamma(2)-\gamma(0))+\gamma(2)(\gamma(0)^2-\gamma(2)^2)+\gamma(1)\gamma(3)(\gamma(2)-\gamma(0))}{\gamma(0)(\gamma(0)^2-\gamma(1)^2)-\gamma(1)^2(\gamma(0)-\gamma(2))+\gamma(2)(\gamma(1)^2-\gamma(0)\gamma(2))}; \\ a_3 &= \tfrac{\gamma(1)(\gamma(1)^2-\gamma(0)\gamma(2))+\gamma(1)\gamma(2)(\gamma(2)-\gamma(0))+\gamma(3)(\gamma(0)^2-\gamma(1)^2)}{\gamma(0)(\gamma(0)^2-\gamma(1)^2)-\gamma(1)^2(\gamma(0)-\gamma(2))+\gamma(2)(\gamma(1)^2-\gamma(0)\gamma(2))}. \end{aligned} \tag{8}$$

Theorem 3.7 If Y_t is fractional Gaussian noise, then linear predictor that uses 3 previous observation $\hat{Y}_{t,3}$ is the best linear predictor if-and-only-if a_1, a_2, and a_3 are equal to the values in expression (8) for $\hat{Y}_{t,3} = a_1 Y_{t-1} + a_2 Y_{t-2} + a_3 Y_{t-3}$

3.3 Comparing minimum mean squared error of linear predictors

Until now the objective was to determine the coefficients for predictors that minimize the mean squared error. Once we derive these coefficients, we can determine the minimum mean squared error (MMSE) of these linear predictors. We can calculate MMSE by using the following theorem [14]:

Theorem 3.8 (MMSE) Suppose $(X_t)_{t\in T}$ is WSS, zero mean stochastic process. Let $n \in \mathbb{N}$ be fixed, then if $\hat{X}_{t,n} := \sum_{i=1}^{n} a_i X_{t-i}$ minimizes $\mathbb{E}(X_t - \hat{X}_{t,n})^2$ over all $a_1, ..., a_n$, the MMSE $\mathbb{E}(X_t - \hat{X}_{t,n})^2$ can be calculated using the following expression:

$$\mathbb{E}(X_t - \hat{X}_{t,n})^2 = \gamma(0) - a_1\gamma(1) - ... - a_n\gamma(n).$$

Because fractional Gaussian noise is WSS and has zero mean Theorem 3.8 can be applied to calculate MMSE for each of the predictors as follows:

$$\mathbb{E}(Y_t - \hat{Y}_{t,1})^2 = \gamma(0) - a_1\gamma(1);$$
$$\mathbb{E}(Y_t - \hat{Y}_{t,2})^2 = \gamma(0) - a_1\gamma(1) - a_2\gamma(2);$$
$$\mathbb{E}(Y_t - \hat{Y}_{t,3})^2 = \gamma(0) - a_1\gamma(1) - a_2\gamma(2) - a_3\gamma(3),$$

where we use coefficients a_i that we determined in the previous subsection (Theorem 3.5, Theorem 3.6, and Theorem 3.7).

By substituting the values of $\gamma(i)$ using the autocovariance expression (1) for fractional Gaussian noise, we get the expressions of MMSE that depend on Hurst parameter H. Therefore, the resulting values of MMSE can be plotted as a function of H:

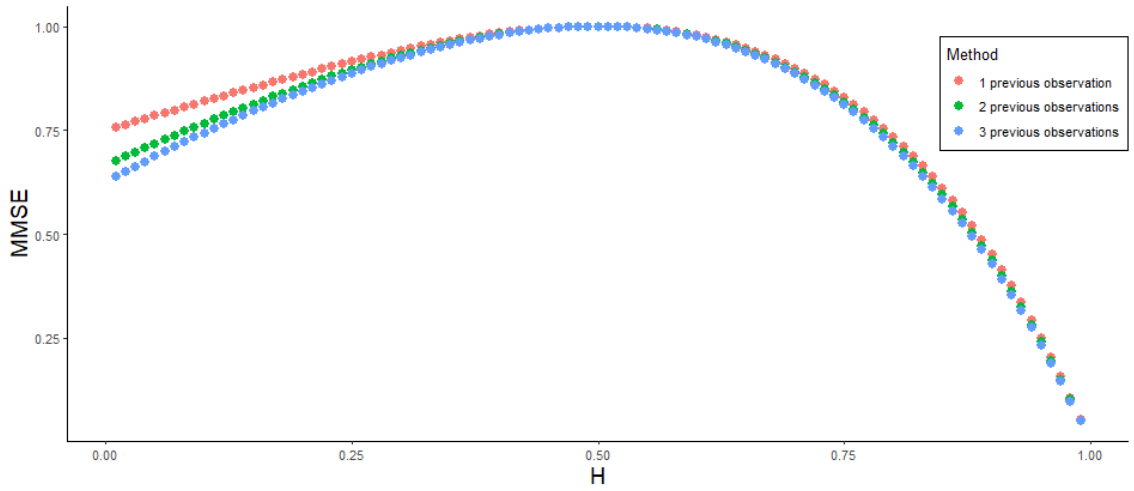

FIGURE 1: MMSE depending on Hurst parameter and number of previous observations.

For fBm processes, the higher the value of Hurst parameter H, the smoother the path of each realization of fBm. Therefore, for higher values of H, the curve of fBm becomes more linear and predictions become more precise and the MMSE approaches 0. While, for lower values of H, the curve of fBm contains a lot of fluctuations, therefore the MMSE values are higher. Lastly, for $H = 0.5$, the fBm process becomes standard white noise, therefore being difficult to predict and constituting the highest MMSE value.

As could be seen in the Figure (1), the MMSE values improve based on the number of previous observations used. However, the improvement is not that significant for higher H values. Additionally, the improvement from using 1 previous observation to using 2 previous observations is more significant than using 3 previous observations rather than 2. This pattern of less significant improvement continues with each additional observation, thus it is enough to use up to 3 previous observations to make predictions.

3.4 Comparing MSE of linear predictors

Another way to compare the MSE of linear predictors is by using Monte Carlo simulations. Using simulations we can check the squared error made through many simulation runs, thus getting MSE. The simulations consisted of firstly simulating 500 values of Fractional Brownian motion X_t for each run and then calculating $Y_t = X_t - X_{t-1}$. Then for a fixed value of H, the squared error $(Y_t - \hat{Y}_{t,n})^2$, where $n = 1, 2, 3$, is calculated for each linear predictor for that run. The coefficients a_i, where $i = 1, 2, 3$, we use in this simulation for $\hat{Y}_{t,n}$ are the ones that minimize the MSE and were derived in Section 3.2. The same process is repeated for 2000 runs for a specific value of H and the mean squared error is calculated. This process is then repeated for all values of $H \in [0.01, 1]$ in increments of 0.01. Plotting the results as a function of Hurst parameter H, we get the following figure:

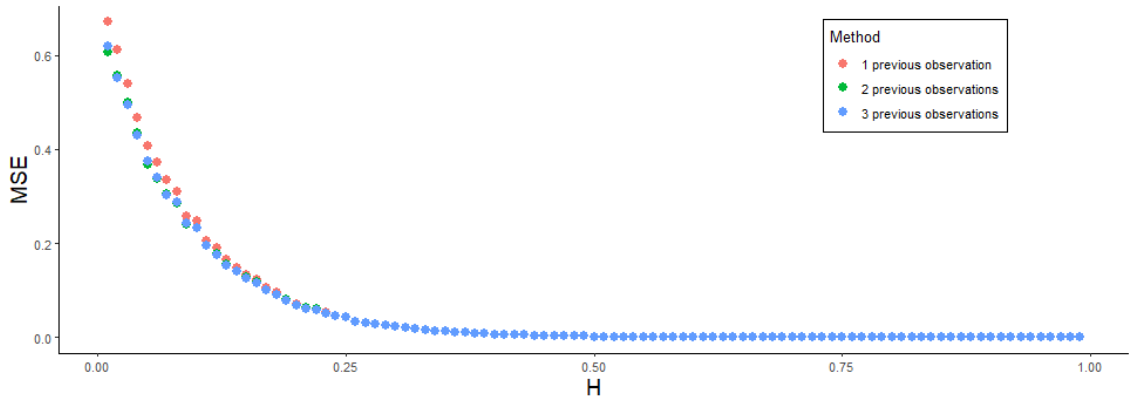

FIGURE 2: MSE depending on Hurst parameter and number of previous observations.

Figure (2) shows that at lower values of H, there is a substantial difference in MSE between using 1 previous observation and using 2 or 3 previous observations to forecast the value of Y_t. However, it can also be noticed that there is barely any difference in MSE when comparing using 2 and 3 previous observations. Additionally, sometimes when considering 2 previous observations, the MSE was lower than when using 3 observations. But for higher values of H, there is barely any difference in MSE and they essentially converge to MSE value near 0. Thus, it can be stated that in terms of MSE, the number of previous observations considered is only favorable for smaller Hurst parameter values.

4 Predictions using Ordinal patterns

Another way to make time-series predictions is by using ordinal patterns. While ordinal patterns 'lose' information about concrete values of X_t and use the relative rank of these values in partition, they can be used to predict whether the next observation will be higher or lower than the previous observation. An article "Short-term prediction through ordinal patterns" by Yair Neuman, Yochai Cohen and Boaz Tamir [5] looks into the use of ordinal patterns to make short-term predictions. By using ordinal patterns we limit the possible transitions to the next ordinal pattern. If we consider ordinal patterns of order 3, then each ordinal pattern, which there is a total of 6 ordinal patterns, can only transition to 3 ordinal patterns. Because of the limitations imposed by the few possible transitions that an ordinal pattern can transition to, it is interesting to analyse how accurate the predictions become if the time series is analysed using ordinal patterns.

In the article [5], the use of ordinal patterns of order 3 was considered and the table containing all possible transitions for ordinal patterns of order 3 can be seen below:

Ordinal pattern π	Possible next ordinal pattern π		
$\pi_1 = (0,1,2)$	$\pi_1 = (0,1,2)$	$\pi_2 = (0,2,1)$	$\pi_4 = (1,2,0)$
$\pi_2 = (0,2,1)$	$\pi_3 = (1,0,2)$	$\pi_5 = (2,0,1)$	$\pi_6 = (2,1,0)$
$\pi_3 = (1,0,2)$	$\pi_1 = (0,1,2)$	$\pi_2 = (0,2,1)$	$\pi_4 = (1,2,0)$
$\pi_4 = (1,2,0)$	$\pi_3 = (1,0,2)$	$\pi_5 = (2,0,1)$	$\pi_6 = (2,1,0)$
$\pi_5 = (2,0,1)$	$\pi_1 = (0,1,2)$	$\pi_2 = (0,2,1)$	$\pi_4 = (1,2,0)$
$\pi_6 = (2,1,0)$	$\pi_3 = (1,0,2)$	$\pi_5 = (2,0,1)$	$\pi_6 = (2,1,0)$

TABLE 1: Possible transitions for ordinal patterns of order 3.

These transitions also can be used to determine whether there was an increase between the two consecutive values ($Y_t > 0$) or a decrease ($Y_t < 0$). This is done by looking at the last two rank values of the next ordinal pattern:

$$\begin{aligned}
\pi_1 = (0,1,2) &\implies Y_t > 0;\\
\pi_2 = (0,2,1) &\implies Y_t < 0;\\
\pi_3 = (1,0,2) &\implies Y_t > 0;\\
\pi_4 = (1,2,0) &\implies Y_t < 0;\\
\pi_5 = (2,0,1) &\implies Y_t > 0;\\
\pi_6 = (2,1,0) &\implies Y_t < 0.
\end{aligned}$$

For example, if the transition was to a pattern $(0,1,2)$, by looking at the rank of the last two observations in the pattern sequence, it can be concluded that the value X_t is higher than X_{t-1}, therefore $Y_t = X_t - X_{t-1}$ results in a positive number. On the contrary, if the transition was to a pattern $(1,2,0)$, then the conclusion would be that X_t is lower than X_{t-1} and thus Y_t is a negative number.

Therefore, by analysing time series using ordinal patterns and counting how many times specific transitions occur, it is possible to predict whether the next value will be higher than the last observed ($sgn(Y_t) = 1$) or lower ($sgn(Y_t) = -1$). This is done by finding the final ordinal pattern in the time series and then examining whether, throughout the entire series, this pattern more frequently led to an increase or a decrease in value.

Let $n_{i,j}$ denote the number of times the transition from π_i to π_j occurred during the entire time series. Then $n_{i,j}$ for time series $(X_t)_{t\in T}$ can be formally expressed as:

$$n_{i,j} = \sum_{t=1}^{T-3} \mathbf{1}\{(X_{t+1}, X_{t+2}, X_{t+3}) = \pi_j | (X_t, X_{t+1}, X_{t+2}) = \pi_i\}.$$

We predict using the last ordinal pattern observed and whether there were more transitions to increasing or decreasing patterns. Table (2) summarizes the predictions being made based on these observations.

Last observed ordinal patterns π	Prediction $sgn(\hat{Y}_t) = 1$ if:	Prediction $sgn(\hat{Y}_t) = -1$ if:
$\pi_1 = (0,1,2)$	$n_{1,1} \geq n_{1,2} + n_{1,4}$	$n_{1,1} < n_{1,2} + n_{1,4}$
$\pi_2 = (0,2,1)$	$n_{2,6} \leq n_{2,3} + n_{2,5}$	$n_{2,6} > n_{2,3} + n_{2,5}$
$\pi_3 = (1,0,2)$	$n_{3,1} \geq n_{3,2} + n_{3,4}$	$n_{3,1} < n_{3,2} + n_{3,4}$
$\pi_4 = (1,2,0)$	$n_{4,6} \leq n_{4,3} + n_{4,5}$	$n_{4,6} > m_{4,3} + m_{4,5}$
$\pi_5 = (2,0,1)$	$n_{5,1} \geq n_{5,2} + n_{5,4}$	$n_{5,1} < n_{5,2} + n_{5,4}$
$\pi_6 = (2,1,0)$	$n_{6,6} \leq n_{6,3} + n_{6,5}$	$n_{6,6} > n_{6,3} + n_{6,5}$

TABLE 2: Predictions made based on last observed ordinal pattern and information from time series.

For example, if the last observed ordinal pattern was π_1 and there were more transitions from π_1 to π_1 then transitions from π_1 to π_2 and π_4 combined, then the prediction would be that the next observed value X_t will be higher than X_{t-1}. On the contrary, if there is more combined transition to patterns π_2 and π_4 than to π_1 then the prediction would be that the next observed value X_t will be lower than X_{t-1}.

Therefore, the ordinal pattern prediction method utilizes the information of the entire time series to distinguish trends in the time series and predict whether the next observed value X_t will be higher or lower than the last observed value X_{t-1}.

5 Comparison of two prediction methods

In this section, we will compare the performance of the ordinal pattern method and the linear predictors for time series forecasting. The comparison entails comparing both methods on a common dataset, followed by a thorough examination of the accuracy of predictions.

Because the ordinal pattern method predicts $sgn(Y_t)$ rather than a concrete value, the linear predictors will also be adjusted to predict $sgn(Y_t)$ rather than a concrete value of Y_t. This is rather straightforward because linear predictor $\hat{Y}_{t,n}$ results in either a positive or negative value. Therefore, taking $sgn(\hat{Y}_{t,n})$ as a prediction for $sgn(Y_t)$ is fitting and allows for an accurate comparison between the two methods.

Comparing the predictions made by both methods to the real $sgn(Y_t)$, it can be determined whether the prediction was correct or incorrect. This can be formulated as a conditional function:

$$\mathbf{1}_{prediction} = \begin{cases} 1 & \text{If the prediction was correct} \\ 0 & \text{If the prediction was incorrect} \end{cases}.$$

Then over many different predictions, we can calculate the accuracy of predictions made using the following expression:

$$\text{Accuracy} = \frac{\sum_N \mathbf{1}_{prdiction}}{N},$$

where N denotes the number of predictions made.

To compare the accuracy of the two prediction methods, we once again simulate 500 values of fractional Brownian motion X_t for each run. Then the entire series is analysed using ordinal patterns, the last occurring pattern is identified and the prediction of $sgn(\hat{Y}_t)$ is made. Similarly, $\hat{Y}_{t,n}$ of the fractional Brownian motion is calculated for $n = 1, 2, 3$ and then $sgn(\hat{Y}_{t,n})$ is the prediction made by linear predictor method using n previous observations. These predictions then are compared to the value of $sgn(Y_t)$ and it is noted whether the prediction was correct or incorrect. The same process is repeated for 2000 runs for a specific value of H and the accuracy of predictions is calculated. This process is then repeated for all values of $H \in [0.01, 1]$ in increments of 0.01. Thus the results can be plotted as a function of Hurst parameter H:

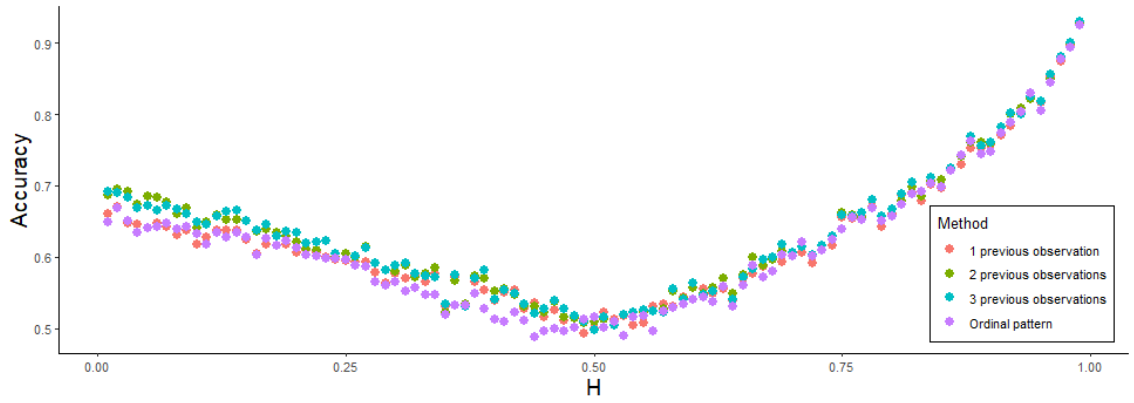

FIGURE 3: Comparison of accuracy for all prediction methods.

When observing Figure (3) it can be seen that the accuracy of predictions made by all of the methods is similar for higher values of H. However, for the lower values of H, the accuracy of predictions made by using ordinal patterns is worse than the predictions made by the linear predictor method that uses 2 or 3 previous observations. Lastly, for H values around 0.5, the accuracy of predictions made by all of the methods is around 0.5. Observe the expression for autocovariance of fGn:

$$\gamma(k) = \frac{1}{2}\left(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}\right).$$

For $H = 0.5$, the expression can be rewritten as:

$$\gamma(k) = \frac{1}{2}\left(|k+1| - 2|k| + |k-1|\right) = \begin{cases} 0 & \text{if } k \neq 0 \\ 1 & \text{if } k = 0 \end{cases}$$

And since for fGn $\mathbb{E}(Y_t) = 0$ for all $t \in \mathbb{N}$, the process becomes standard white noise and thus predictions become random guessing constituting to accuracy around 0.5.

6 Hurst estimator, Moving average processes, and Autoregressive processes

This section investigates an approach for estimating the Hurst parameter for any wide-sense stationary (WSS) process. Furthermore, we describe two new WSS linear processes: Moving Averages (MA) and Autoregressive (AR), and then test and compare the accuracy of previously developed prediction methods on these processes.

6.1 Hurst estimator

In the theoretical setting, the Hurst parameter is known and the data set is generated using that Hurst parameter. Therefore comparing the accuracy of predictions made by the 2 prediction methods is rather subjective. When it comes to making predictions for real-world data, the Hurst parameter H is unknown and thus predicting the next value using a linear predictor method becomes a difficult task. However, there is a way to estimate the Hurst parameter for any wide-sense stationary (WSS) data using ordinal patterns. The article "Estimation of ordinal pattern probabilities in fractional Brownian motion" by Mathieu Sinn and Karsten Keller [9], analyses how the Hurst parameter can be estimated using ordinal patterns for any WSS time series.

In the article "Ordinal patterns in long-range dependent time series" by A. Betken et al. [16], the idea behind Hurst parameter estimation is to analyse the "up-and-down" behavior of the time series. For ordinal patterns, this entails analysing which ordinal patterns constitute fluctuating rank values, rather than increasing or decreasing rank values. For ordinal patterns of order 3, we formulate this statement as a conditional function:

$$W(i) := \mathbf{1}_{\{\pi(X_i,X_{i+1},X_{i+2})\}} = \begin{cases} 1 & \text{if } \pi(X_i, X_{i+1}, X_{i+2}) \in \{(2,0,1),(1,0,2),(0,2,1),(1,2,0)\} \\ 0 & \text{else} \end{cases}$$

Then the relative frequency estimator is given by:

$$\hat{c}_n = \frac{1}{n}\sum_{i=1}^{n} W(i). \tag{9}$$

We can use this relative frequency $\hat{c}_n$ for the Zero-crossing estimator of the Hurst parameter [16]. The estimator of Hurst parameter can be calculated using the following expression:

$$\hat{H}_n := max\left\{0, log_2\left(cos\left(\frac{\pi \hat{c}_n}{2}\right)\right) + 1\right\},$$

where $\hat{c}_n$ has a value calculated using expression (9). It should be noted that this estimator of the Hurst parameter only applies when considering fBm data.

To showcase how accurate the Hurst parameter estimator $\hat{H}$ is, we can simulate values for fractional Brownian motion using real Hurst parameter H. Then analysing the time series using ordinal patterns, calculate the estimator of Hurst parameter $\hat{H}$ and find an error between the real and estimated values of the Hurst parameter. The simulation was done for all values of $H \in [0.01, 1]$ in increments of 0.01 and then was plotted as a function of the real Hurst parameter H. The resulting plot can be seen in Figure (4).

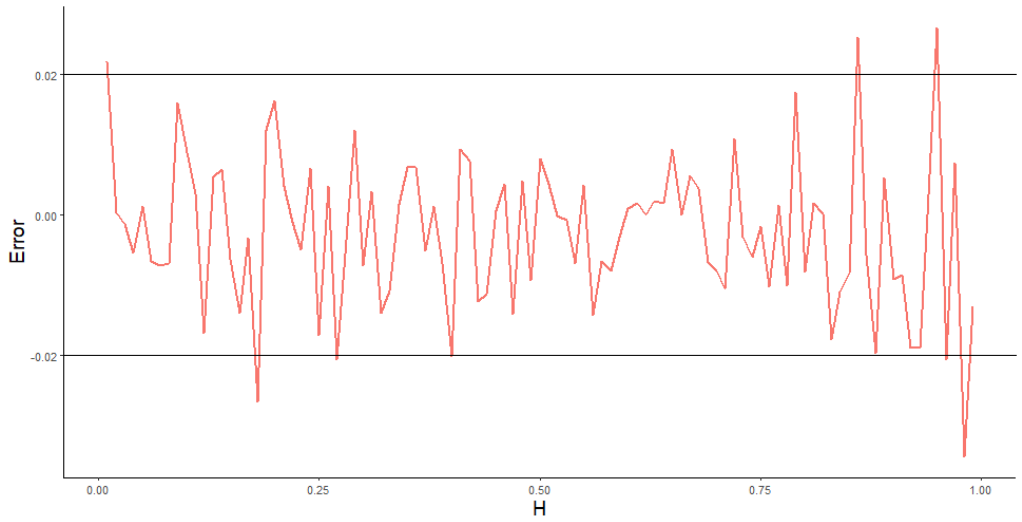

FIGURE 4: The error between the estimated ($\hat{H}$) and real (H) Hurst parameter.

As could be seen, the difference is not really substantial with most of the errors fluctuating between -0.02 and 0.02, meaning that the estimator of Hurst parameter $\hat{H}$ is an accurate alternative that can be used for other WSS processes.

6.2 Moving average processes

Until now, we only considered using fractional Brownian motion processes. However, there are many other linear processes that are WSS. One example of a WSS linear process is a Moving averages process. A moving averages process of order k is a linear process X_t that can be described using the following equation:

$$X_t = b_0\epsilon_t + b_1\epsilon_{t-1} + \cdots + b_k\epsilon_{t-k},$$

where $b_i \in \mathbb{R}$, $i = 0, 1, \ldots, k$ and ϵ_t is white noise. The process is the weighted sum of $k+1$ preceding white noise values. For convenience, we denote a moving averages process of order k by MA(k).

Consider a MA(1) process. Then each X_t can be expressed using the following equation:

$$X_t = b_0\epsilon_t + b_1\epsilon_{t-1}.$$

Then we can compare the predictions made by the two methods using Monte Carlo simulations. For simulations, we let $b_0 = 1$ and $b_1 = \frac{a}{2}$. Then we simulate values X_t for values of $a \in [0, 1]$ in increments of 0.1. Then the predictions for each run are made by linear predictor that uses 1 previous observation (LP(1) method) and ordinal pattern method (OP method). Then the accuracy is calculated over the 10000 simulation runs. The results of this simulation can be seen in Table (3).

MA(1)	Accuracy	
Value of a	OP method	LP(1) method
$a = 0$	0.5	0.5
$a = 0.1$	0.51	0.5
$a = 0.2$	0.52	0.53
$a = 0.3$	0.52	0.53
$a = 0.4$	0.53	0.54
$a = 0.5$	0.54	0.56
$a = 0.6$	0.55	0.58
$a = 0.7$	0.55	0.59
$a = 0.8$	0.58	0.61
$a = 0.9$	0.58	0.61
$a = 1$	0.59	0.63

TABLE 3: Accuracy of predictions of simulated MA(1) process.

Similarly, we can consider MA(2) process, with the following expression:

$$X_t = \epsilon_t + \frac{a}{2}\epsilon_{t-1} + \frac{a}{4}\epsilon_{t-2}.$$

Once again we can compare the accuracy of predictions made by LP(1), LP(2), and ordinal pattern method through simulations. The accuracy of predictions made for each value of $a \in [0, 1]$ is shown in the table below.

MA(2)	Accuracy		
Value of a	OP method	LP(1) method	LP(2) method
$a = 0$	0.5	0.5	0.49
$a = 0.1$	0.5	0.5	0.5
$a = 0.2$	0.51	0.51	0.52
$a = 0.3$	0.53	0.53	0.53
$a = 0.4$	0.55	0.55	0.55
$a = 0.5$	0.56	0.58	0.58
$a = 0.6$	0.58	0.6	0.6
$a = 0.7$	0.59	0.62	0.61
$a = 0.8$	0.6	0.62	0.62
$a = 0.9$	0.61	0.63	0.63
$a = 1$	0.64	0.67	0.66

TABLE 4: Accuracy of predictions of simulated MA(2) process.

Lastly, consider the following expression for the MA(3) process:

$$X_t = \epsilon_t + \frac{a}{2}\epsilon_{t-1} + \frac{a}{4}\epsilon_{t-2} + \frac{a}{8}\epsilon_{t-3}.$$

In simulations, the predictions are made by using LP(1), LP(2), LP(3), and ordinal pattern method and the resulting accuracy can be seen in Table (5).

MA(3)	Accuracy			
Value of a	OP method	LP(1) method	LP(2) method	LP(3) method
$a = 0$	0.5	0.49	0.49	0.49
$a = 0.1$	0.51	0.51	0.51	0.51
$a = 0.2$	0.52	0.52	0.52	0.52
$a = 0.3$	0.54	0.54	0.53	0.53
$a = 0.4$	0.54	0.56	0.56	0.56
$a = 0.5$	0.55	0.57	0.57	0.57
$a = 0.6$	0.57	0.6	0.6	0.6
$a = 0.7$	0.59	0.61	0.62	0.61
$a = 0.8$	0.61	0.63	0.63	0.63
$a = 0.9$	0.63	0.65	0.65	0.64
$a = 1$	0.65	0.67	0.67	0.66

TABLE 5: Accuracy of predictions of simulated MA(3) process.

Multiple conclusions can be drawn by looking at the accuracy of all of the moving average process simulations. Firstly, the accuracy of predictions made using all of the methods were similar. Secondly, the accuracy of all predictions increased with the increasing value of a. This was to be expected because by increasing the value of a the realizations of the moving averages process becomes more linear. Lastly, when we consider $a = 0$, all of the considered MA processes become just a single white noise observation and therefore like discussed in section (5), the autocovariance $\gamma(k) = 1$ when $k = 0$ and $\gamma(k) = 0$ for $k \neq 0$. This indicates that methods work as intended because any accuracy substantially higher than 0.5 would contradict the fact that white noise is unpredictable. Additionally, it should be noted that we do not use LP(2) and LP(3) prediction methods for the MA(1) process and LP(3) for the MA(2) process, since the values of these processes depend on 1 and 2 previous ϵ values, therefore these predictions methods would not be a valid choice.

6.3 Autoregressive processes

Another example of a linear process that can be WSS is the autoregressive process. An autoregressive process of order k is a linear process X_t that is described by the following equation:

$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_k X_{t-k} + \epsilon_t,$$

where $a_i \in \mathbb{R}$, $i = 1, 2, \ldots, k$ and ϵ_t is white noise. The value of X_t is the weighted sum of k preceding X observations and white noise ϵ_t. For convenience, we denote an autoregressive process of order k as AR(k).

If we consider the AR(k) process as a polynomial of degree k, then the AR(k) process is asymptotic, if all the roots of this polynomial lie inside the unit circle [11]. For simulations, we consider AR processes that are WSS.

Once again we can compare the accuracy of predictions made by LP(1), LP(2), LP(3), and ordinal pattern prediction methods using Monte Carlo simulations that we applied for moving averages processes. In this case for AR(1), AR(2), and AR(3) processes we can use all prediction methods, because the autoregressive process is recurring, meaning that all previous values in a time series influence the next value. We consider the following AR(1),

AR(2), and AR(3) processes:

$$\textbf{AR(1) process: } X_t = \frac{a}{2}X_{t-1} + \epsilon_t;$$
$$\textbf{AR(2) process: } X_t = \frac{a}{2}X_{t-1} + \frac{a}{4}X_{t-2} + \epsilon_t;$$
$$\textbf{AR(3) process: } X_t = \frac{a}{2}X_{t-1} + \frac{a}{4}X_{t-2} + \frac{a}{8}X_{t-3} + \epsilon_t.$$

This choice of coefficients ensures, that all of the autoregressive processes are WSS for values of $a \in [0, 1]$. The resulting accuracy of predictions made are summarized in the tables below:

$AR(1)$	Accuracy			
Value of a	OP method	LP(1) method	LP(2) method	LP(3) method
$a = 0$	0.51	0.51	0.5	0.5
$a = 0.1$	0.51	0.51	0.5	0.51
$a = 0.2$	0.51	0.52	0.53	0.52
$a = 0.3$	0.52	0.52	0.52	0.52
$a = 0.4$	0.53	0.55	0.55	0.55
$a = 0.5$	0.55	0.57	0.57	0.57
$a = 0.6$	0.57	0.58	0.58	0.58
$a = 0.7$	0.58	0.6	0.6	0.59
$a = 0.8$	0.6	0.63	0.62	0.62
$a = 0.9$	0.62	0.65	0.64	0.64
$a = 1$	0.64	0.66	0.66	0.66

TABLE 6: Accuracy of predictions of simulated AR(1) process.

$AR(2)$	Accuracy			
Value of a	OP method	LP(1) method	LP(2) method	LP(3) method
$a = 0$	0.51	0.5	0.5	0.5
$a = 0.1$	0.51	0.51	0.51	0.51
$a = 0.2$	0.52	0.51	0.52	0.52
$a = 0.3$	0.54	0.54	0.54	0.54
$a = 0.4$	0.55	0.55	0.56	0.56
$a = 0.5$	0.58	0.58	0.59	0.59
$a = 0.6$	0.6	0.61	0.62	0.61
$a = 0.7$	0.61	0.63	0.64	0.63
$a = 0.8$	0.66	0.66	0.67	0.67
$a = 0.9$	0.69	0.7	0.71	0.71
$a = 1$	0.72	0.73	0.73	0.73

TABLE 7: Accuracy of predictions of simulated AR(2) process.

$AR(3)$	Accuracy			
Value of a	OP method	LP(1) method	LP(2) method	LP(3) method
$a = 0$	0.5	0.5	0.5	0.51
$a = 0.1$	0.5	0.5	0.51	0.51
$a = 0.2$	0.52	0.52	0.52	0.52
$a = 0.3$	0.54	0.54	0.54	0.54
$a = 0.4$	0.55	0.56	0.57	0.57
$a = 0.5$	0.58	0.59	0.6	0.6
$a = 0.6$	0.61	0.62	0.63	0.63
$a = 0.7$	0.64	0.66	0.66	0.67
$a = 0.8$	0.68	0.68	0.69	0.7
$a = 0.9$	0.73	0.73	0.75	0.74
$a = 1$	0.79	0.79	0.8	0.8

TABLE 8: Accuracy of predictions of simulated AR(3) process.

Once again, the accuracy of predictions made using all of the methods were similar. Additionally, when observing the results of different AR(k) processes, it can be seen that the accuracy of predictions increased faster when more previous observations were considered for the AR process. This is because increasing the number of observations considered for AR processes, for higher values of a constituted fewer fluctuations in time series. Once, again in all the simulations for $a = 0$, the AR processes resulted in single white noise observation, and therefore the accuracy of these predictions being approximately 0.5 indicates, that the methods work as intended.

7 Predictions for real-world data

This section compares time series prediction methods using two real-world datasets: oil prices and global temperatures. We aim to evaluate the performance and accuracy of these methods in predicting trends in each dataset.

7.1 Oil prices

As a data set, we consider crude oil prices of Western Texas intermediate. This data set contains 10021 daily prices in dollars per barrel that span from January of 1986 until the end of May 2024. Some data points were missing or had a value of 0$ per barrel, therefore to be able to represent it as a continuous and accurate time series, the values were taken as a weighted average of the two closest non-zero values. The plot depicting the price change over time (time series) can be seen in Figure (5).

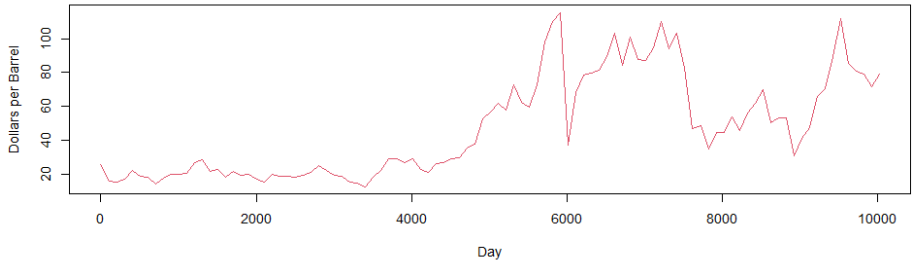

FIGURE 5: Time series of daily oil price from 1986 until now.

However, the time series is not WSS and needs to be processed so that the model assumptions are satisfied and predictions using linear predictors can be made. In the article "Order patterns, their variation and change points in financial time series and Brownian motion" by Christoph Bandt [10], the same data set is used, and logarithmic returns of the prices are taken to make the data WSS. The logarithmic returns for daily data are calculated as:

$$R_t = log(X_t) - log(X_{t-1}),$$

where R_t denotes the logarithmic return of day t. The resulting time series contains 10020 logarithmic return values, which can be seen in Figure (6):

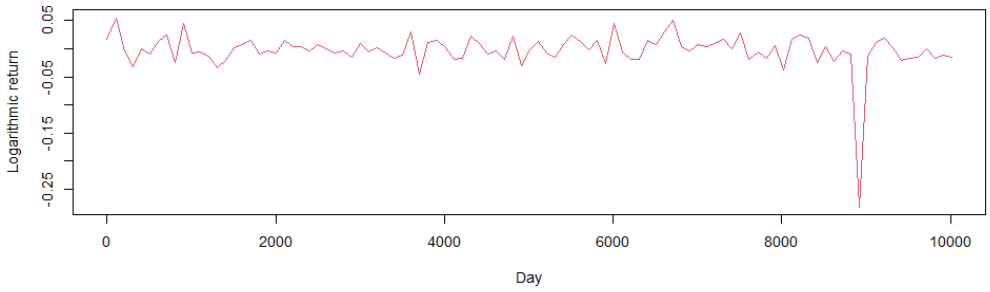

FIGURE 6: Time series of logarithmic returns of oil prices.

Inspecting this plot, it could be seen that there is a large decrease followed by a large increase in the time series around day 9000. This spike occurred between the start of march 2020 and the start of April 2020, when there was a global pandemic Covid-19. The fast decrease, was a result of the lockdown that occurred during the start of the pandemic, when the demand for oil decreased rapidly. The quick increase was the result of economic recovery and supply disruptions after the lockdown. Therefore, we can see that global events impact the prices of oil and there is dependence in time series data. When observing the logarithmic return values over a shorter period, see Figure (7), the values fluctuate a lot and showcase the random behavior of the observations.

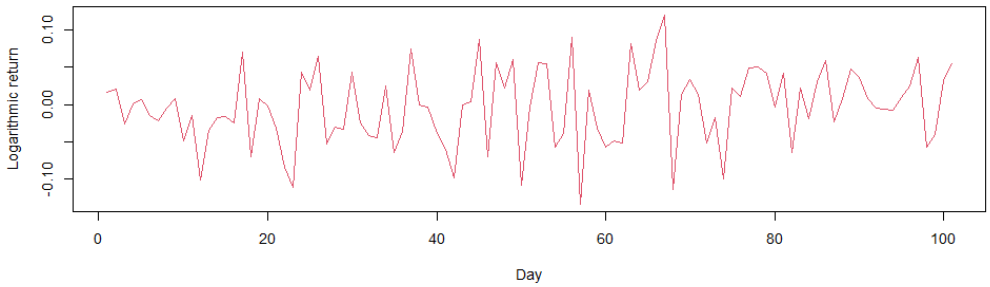

FIGURE 7: Smaller time series of logarithmic returns of oil prices.

To be able to make predictions, we need to show that the time series $(R_t)_{t\in T}$ satisfies the model assumptions of having zero mean $\mathbb{E}(R_t) = 0$ for all $t \in N$ and being WSS. Calculating the sample mean of the time series, the derived result was 0.0001. Therefore, time series $(R_t)_{t\in T}$ has a sample mean of nearly 0 and this condition is satisfied.

To see if the time series is stationary, we can analyse the resulting autocovariance at lag k. To conclude that the time series is WSS, the resulting autocovariance should be 1 at lag 0 and around 0 for the remaining lag k, where $k \geq 1$. Calculating the autocovariance at specific lag k and plotting the results we get the following:

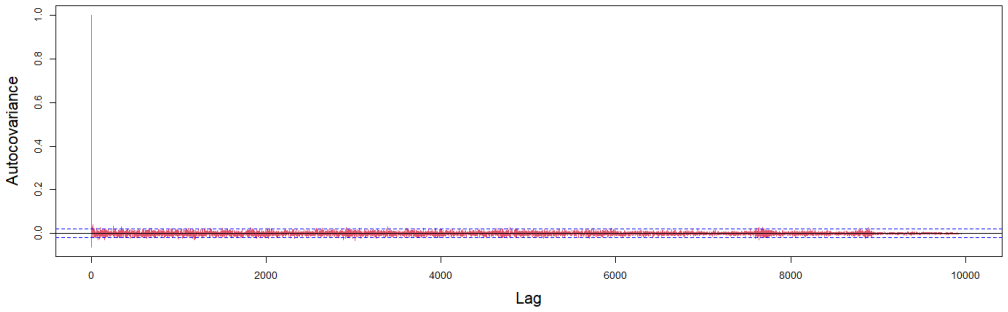

FIGURE 8: The autocovariance at specific lag for time series R_t

As could be observed in Figure (8), $\gamma(0) = 1$ and the values of $\gamma(k)$ for $k \neq 0$ are close to 0. Therefore, we can say that the time series $(R_t)_{t \in T}$ satisfies the model assumptions, and can be used to make predictions.

For predictions, we split the time series of 10020 logarithmic return values into smaller overlapping time series of 102 values. So we first consider the values from day 1 until day 102, then from day 2 until 103, and so on. For each interval, we aim to predict the sign of the last logarithmic return in that interval. If the sign is positive this indicates that the oil price was higher than the previous day. On the contrary, if the sign is negative, it indicates that oil prices decreased. For each interval, we use the remaining 101 known values of logarithmic returns, to estimate the Hurst parameter H. Then, the predictions are made using the linear predictor method and the ordinal pattern method and are compared to the real $sgn(R_t)$. Overall, during the entire process, this constituted to 9199 intervals and thus 9199 total predictions.

Accuracy			
OP method	LP(1) method	LP(2) method	LP(3) method
0.505	0.511	0.502	0.499

TABLE 9: Accuracy of predictions for logarithmic return values.

As could be seen in Table (9), the accuracy of all prediction methods is around 0.5, which is similar to an accuracy one would achieve if they were to randomly guess. However, these were the expected results, because if the accuracy of any of the methods were substantially better than 0.5, it could be concluded that the oil prices are predictable and people could use these methods to benefit financially. Therefore, this data set confirms that these methods work as intended and thus is theoretical confirmation of the prediction methods considered.

7.2 Global temperatures

Another data set we consider is the monthly global world temperatures. This data set contains 1392 monthly global average temperatures that span from January 1900 until December 2015. This is the resulting plot, that depicts the average temperature change over time.

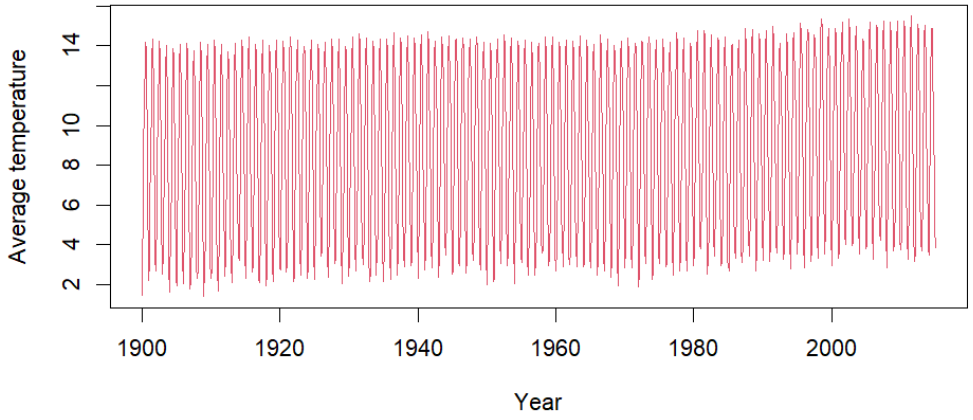

FIGURE 9: Time series of average global temperature from 1900 until 2015.

When observing Figure (9), it can be concluded that the resulting time series is not WSS, because we can see a lot of fluctuations, the average temperature gradually increasing as years go by. Therefore, the time series first needs to be preprocessed, before applying the prediction methods. Time series can be separated into three different components [14]:

- A trend m_t, which showcases the gradual development of time series.
- A seasonal component s_t, which showcases the pronounced periodic behaviour of time series.
- An incidental component w_t, which showcases the irregular fluctuations of time series.

By decomposing the global temperature time series into these 3 components, we get the following plots:

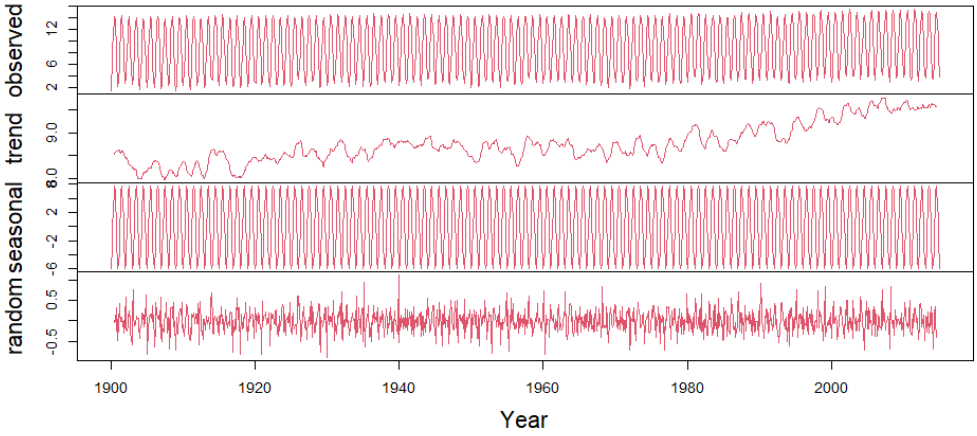

FIGURE 10: Decomposition of average global temperature time series.

Firstly, when observing the trend component of the time series in Figure (10), we can see that the temperature gradually increased over the observed period. Additionally, this increase becomes steeper from Year 1980 and continues until the end of the time series. This perfectly corresponds to what we already know about global temperature and global warming. Lastly, the incidental component, or in the plot the random component, corresponds to the difference from the expected global temperature that month. Therefore, by predicting whether the random component will be a positive or a negative value, we would be able to determine whether the global temperature will be higher or lower than expected for the next month.

We consider the random component in Figure (10) as our time series. Once again, to be able to make a prediction, we need to show that the time series satisfies the model assumptions of having zero mean $\mathbb{E}(X_t) = 0$ and being WSS. Calculating the sample mean of the time series, the derived result was 0.0018. Therefore, the time series has a sample mean of nearly 0 and this condition is satisfied.

Now we analyse the autocovariance at lag k to determine if the time series is stationary and conclude whether the time series satisfies the model assumptions and can be used for predictions. The resulting plot can be seen below:

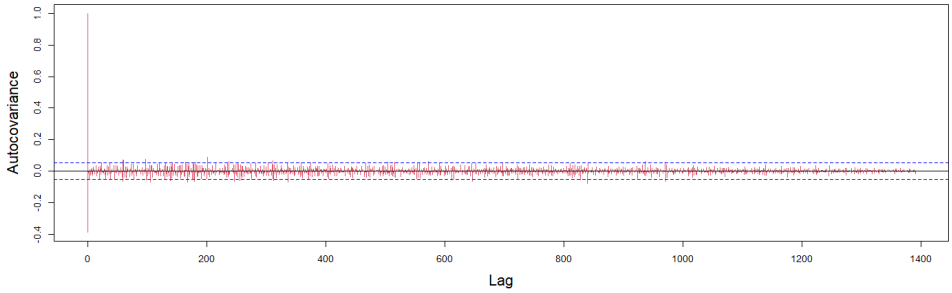

FIGURE 11: The autocovariance at specific lag for the incidental component

Once again, we see in Figure (11), that $\gamma(0) = 1$ and the values of $\gamma(k)$ for $k \neq 0$ are close to 0 thus, we can say that the time series satisfies the model assumptions, and can be used to make predictions.

For predictions, we split the time series of 1392 values into smaller overlapping time series of 120 values. For each interval, we aim to predict the sign of the last observation in that interval. If the sign is positive this indicates that the global temperature that month was higher than expected. On the contrary, if the sign is negative, it indicates that the global temperature was lower than expected. For each interval, we use the remaining 119 known incidental component values of global temperature, to estimate the Hurst parameter H. Then, the predictions are made using the linear predictor method and ordinal pattern predictions method and compared to the real $sgn(X_t)$. Overall, during the entire process, this constituted to 1272 interval and thus 1272 total predictions.

Accuracy			
OP method	LP(1) method	LP(2) method	LP(3) method
0.608	0.614	0.623	0.627

TABLE 10: Accuracy of predictions for incidental component of global temperature

As could be observed in Table (10), the accuracy of all prediction methods is around 0.62, where the highest accuracy was achieved when using 3 previous observations. Additionally, by using more previous observations, the accuracy of predictions increased, which indicates that the incidental component of global temperatures is dependent on past observations. Lastly, because the accuracy is above 0.5, this indicates that the incidental component can be predicted using past data. These methods can be used to forecast whether a global temperature upcoming month will be hotter or colder than expected.

8 Conclusion

In this research, we compared the predictions made by using ordinal patterns (4) and linear predictors (3) for many different stochastic processes. The analysis was conducted on the time series that represented: fractional Brownian motion processes (5), Moving averages processes (6.2), and Autoregressive processes (6.3). Additionally, both methods were tested using real-world data, such as oil prices (7.1) and global temperatures (7.2).

The results showcased that ordinal patterns can be used as a prediction method and in most cases produce better accuracy than 0.5. Under specific circumstances, the predictions made using ordinal patterns can be as accurate as predictions made by the linear predictor method. This was the case when considering fractional Brownian motion processes with $H \geq 0.5$. However, the linear predictor method performed substantially better for lower values of H. This can be seen when analysing results of Moving averages and autoregressive processes, where during simulations the estimated Hurst parameter was between 0 and 0.3.

Additionally, it should be noted, that ordinal patterns proved to be not only a good forecasting method, but also a tool to analyse time series. Ordinal patterns proved to be a great way to estimate the Hurst parameter for any time series, thus allowing to analyse any stochastic process as fractional Brownian motion processes.

In future research, other prediction methods or machine learning can be considered and compared to the ordinal pattern prediction method. Additionally, it could be checked theoretically whether there are any restrictions that limit the accuracy of predictions made using ordinal patterns.

References

[1] C. Bandt and B. Pompe, 'Permutation Entropy: A Natural Complexity Measure for Time Series', Phys. Rev. Lett., vol. 88, no. 17, p. 174102, Apr. 2002, doi: 10.1103/PhysRevLett.88.174102.

[2] K. Keller, A. Unakafov, and V. Unakafova, 'Ordinal Patterns, Entropy, and EEG', Entropy, vol. 16, no. 12, pp. 6212–6239, Nov. 2014, doi: 10.3390/e16126212.

[3] M. Muñoz-Guillermo, 'Ordinal Patterns in Heartbeat Time Series: An Approach Using Multiscale Analysis', Entropy, vol. 21, no. 6, p. 583, Jun. 2019, doi: 10.3390/e21060583.

[4] K. Kalpakis et al., 'Permutation entropy analysis of vital signs data for outcome prediction of patients with severe traumatic brain injury', Computers in Biology and Medicine, vol. 56, pp. 167–174, Jan. 2015, doi: 10.1016/j.compbiomed.2014.11.007.

[5] Y. Neuman, Y. Cohen, and B. Tamir, 'Short-term prediction through ordinal patterns', R. Soc. open sci., vol. 8, no. 1, p. 201011, Jan. 2021, doi: 10.1098/rsos.201011.

[6] P. J. Brockwell and R. A. Davis, Time series: theory and methods, 2nd ed. in Springer series in statistics. New York: Springer, 1996.

[7] W. W. S. Wei, Time Series Analysis: Univariate and Multivariate Methods, 2nd ed. Pearson College Div 2005.

[8] S. Shi, J. Yu, and C. Zhang, 'Fractional gaussian noise: Spectral density and estimation methods', Journal Time Series Analysis, p. jtsa.12750, May 2024, doi: 10.1111/jtsa.12750.

[9] M. Sinn and K. Keller, 'Estimation of ordinal pattern probabilities in fractional Brownian motion'. arXiv, Jan. 10, 2008, arXiv: 0801.1598

[10] C. Bandt, 'Order patterns, their variation and change points in financial time series and Brownian motion', Stat Papers, vol. 61, no. 4, pp. 1565–1588, Aug. 2020, doi: 10.1007/s00362-020-01171-7.

[11] O. Taussky, 'Positive-definite matrices and their role in the study of the characteristic roots of general matrices', Advances in Mathematics, vol. 2, no. 2, pp. 175–186, Jun. 1968, doi: 10.1016/0001-8708(68)90020-0.

[12] S. Brakken-Thal, 'Gershgorin's Theorem for Estimating Eigenvalues'.

[13] R. Bhattacharya and E. C. Waymire, A Basic Course in Probability Theory. in Universitext. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-47974-3.

[14] H. Kwakernaak, G. Meinsma, and A. Betken, Time series analysis, University of Twente: Enschede, The Netherlands, 2023.

[15] M. Lachut and B. Keigwin, 'Applications of the Spectral Theorem: Utilizing Eigenvalues and Eigenvectors', J Stud Res, vol. 12, no. 4, Nov. 2023, doi: 10.47611/jsrhs.v12i4.5585.

[16] Betken A, Buchsteiner J, Dehling H, Münker I, Schnurr A, Woerner JH. Ordinal patterns in long-range dependent time series. Scand J Statist. 2021;48:969–1000. https://doi.org/10.1111/sjos.12478

Appendix

Appendix A: Inverse of 3×3 matrix

Consider any 3×3 matrix A, defined as follows:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}.$$

The inverse of A can be expressed as follow:

$$A^{-1} = \frac{1}{det(A)} adj(A),$$

where $det(A)$ denotes the determinant of matrix A and $adj(A)$ denotes the adjoint matrix of A. Where the determinant and adjoint of matrix A have the following expressions:

$$det(A) = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix};$$

$$adj(A) = \begin{bmatrix} \begin{vmatrix} e & f \\ h & i \end{vmatrix} & -\begin{vmatrix} b & c \\ h & i \end{vmatrix} & \begin{vmatrix} b & c \\ e & f \end{vmatrix} \\ -\begin{vmatrix} d & f \\ g & i \end{vmatrix} & \begin{vmatrix} a & c \\ g & i \end{vmatrix} & -\begin{vmatrix} a & c \\ d & f \end{vmatrix} \\ \begin{vmatrix} d & e \\ g & h \end{vmatrix} & -\begin{vmatrix} a & b \\ g & h \end{vmatrix} & \begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{bmatrix},$$

where:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

is the difference of the product of the diagonals. Therefore we have:

$$det(A) = a(ei - fh) - b(di - fg) + c(dh - eg);$$

$$adj(A) = \begin{bmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix}.$$

Therefore the inverse of the 3×3 matrix can be expressed as the following:

$$A^{-1} = \frac{1}{a(ei - fh) - b(di - fg) + c(dh - eg)} \begin{bmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix},$$

where condition that $a(ei - fh) - b(di - fg) + c(dh - eg) \neq 0$ must be satisfied for matrix A to be invertable.

Appendix B: R studio codes

Appendix B presents all R studio codes that were constructed to be used in this Thesis.

B.1 Script 1

This script was used to calculate and plot the MMSE of linear predictor for fBm process in Section 3.3.

```
library("ggplot2")
#We set starting value of H
H <- 0.01

#We create a vector to store the MMSE values for 1,2 and 3 previous observations
MMSE1 <- c()
MMSE2 <- c()
MMSE3 <- c()

#We run a loop until H = 0.99 in increments of 0.01
while (H < 1){
  #We calculate the MMSE and store the value

  MMSE1 <- append(MMSE1,2 ** (2*H) - 2**(4*H-2))

  MMSE2 <- append(MMSE2,1 - ((2 **(2*H-1) - 1)
  *(1-(3**(2*H)+1)/2 + 2**(2*H))/(2 ** (2*H) - 2**(4*H-2)))*(2**(2*H-1)-1)
  - ((3**(2*H)+1)/2 - 2**(2*H) - (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))
  *((3**(2*H)+1)/2 - 2**(2*H)))

  MMSE3 <- append(MMSE3, 1 - ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)
  *((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H) - 1)
  + (2**(4*H-1)+2**(2*H-1)-3**(2*H))*((2**(2*H-1)-1)**2
  - ((3**(2*H)+1)/2 - 2**(2*H))))/(1-(2**(2*H-1)-1)**2
  - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))
  +((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))
  *(2**(2*H-1)-1) -  (((2**(2*H-1)-1)**2)*((3**(2*H)+1)/2-2**(2*H)-1)
  +((3**(2*H)+1)/2 - 2**(2*H))*(1-((3**(2*H)+1)/2 - 2**(2*H))**2)+(2**(2*H-1)-1)
  *(2**(4*H-1)+2**(2*H-1)-3**(2*H))*((3**(2*H)+1)/2-2**(2*H)-1))
  /(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))
  +((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))
  *((3**(2*H)+1)/2 - 2**(2*H))  - ((2**(2*H-1)-1)*((2**(2*H-1)-1)**2
  - ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)*((3**(2*H)+1)/2
  - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H)-1)+(2**(4*H-1)+2**(2*H-1)-3**(2*H))
  *(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)
  *(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))
  *((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))
  *(2**(4*H-1)+2**(2*H-1)-3**(2*H)))

  H <- H + 0.01
}

#Creating a data frame that contains all the MMSE
df1 <- data.frame(Method=rep(c("1 previous observation","2 previous observations"
,"3 previous observations"),each=99)
,H=rep(c(seq(0.01,0.99, by=0.01)),3),MMSE = c(MMSE1,MMSE2,MMSE3))

#Creating a plot of this Data frame
MMSEplot <- ggplot(df1, aes(x=H, y=MMSE, group=Method, linetype = Method)) +
  geom_point(aes(color=Method),size = 3) +
  theme_classic() +
  theme(
    legend.position = c(0.91, 0.8),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.box.background = element_rect(color = "black"),
    legend.box = "vertical",  # Make the legend vertical
    axis.title.x = element_text(size = 16),
    axis.title.y = element_text(size = 16)
  ) +
  labs(
    x = "H",
    y = "MMSE",
    color = "Method",
    linetype = "Method"
  )


# Print the plot
print(MMSEplot)
```

B.2 Script 2

This script was used to calculate and plot the MSE of linear predictor for fBm process in Section 3.4 and compare the accuracy of all prediction methods for fBm process in Section 5.

```
install.packages("longmemo")
library(longmemo)
install.packages("pracma")
library(pracma)
library(ggplot2)


#Simulation of fractional Brownian motion with Hurst parameter H of length n and wit

circFBM<- function(n, H, plotfBm=FALSE){
  if(missing(n)) n <- 500
  if(missing(H))
    H <- 0.6
  if(missing(plotfBm)) plotfBm <- 1
  ## ------------------------------------------------------------------
  ## first line of the circulant matrix, C, built via covariances of fGn
  ## ------------------------------------------------------------------
  lineC <- function(n, H, m){
    k <- 0:(m - 1)
    H2 <- 2 * H
    v <- (abs((k - 1)/n)^H2 - 2 * (k/n)^H2 + ((k + 1)/n)^H2)/2
    ind <- c(0:(m/2 - 1), m/2, (m/2 - 1):1)
    v <- v[ind + 1]
    drop(v)
  }
  ##
  ## ---------------------
  ## next power of two > n
  ## ---------------------
  m <- 2
  repeat {
    m <- 2 * m
    if(m >= (n - 1))
      break
  }
  stockm <- m      ##
  ## ------------------------------------------------------
  ## research of the power of two (<2^18) such that
  ## C is definite positive
  ## ------------------------------------------------------
  repeat {
    m <- 2 * m
    eigenvalC <- lineC(n, H, m)
    if (m<100) print(eigenvalC)
    eigenvalC <- Re(fft(c(eigenvalC), inverse = F))
    if((all(eigenvalC > 0)) | (m > 2^18))
      break
  }
  if(m > 2^18) {
    cat("----> exact method, impossible!!", fill = T)
    cat("----> can't find m such that C is definite positive", fill
        = T)
    break
  }
  else {
    ##
    ## ------------------------------------------------------
    ## simulation of W=(Q)^t Z, where Z leads N(0,I_m)
    ## and  (Q)_{jk} = m^(-1/2) exp(-2i pi jk/m)
    ## ------------------------------------------------------
    ar <- rnorm(m/2 + 1)
    ai <- rnorm(m/2 + 1)
    ar[1] <- sqrt(2) * ar[1]
    ar[(m/2 + 1)] <- sqrt(2) * ar[(m/2 + 1)]
    ai[1] <- 0
    ai[(m/2 + 1)] <- 0
    ar <- c(ar[c(1:(m/2 + 1))], ar[c((m/2):2)])
    aic <-  - ai
    ai <- c(ai[c(1:(m/2 + 1))], aic[c((m/2):2)])
    W <- complex(real = ar, imaginary = ai) ##
    ## ---------------------
    ## reconstruction of the fGn
    ## ---------------------
    W <- (sqrt(eigenvalC)) * W
    fGn <- fft(W, inverse = F)
    fGn <- (1/(sqrt(2 * m))) * fGn
    fGn <- Re(fGn[c(1:n)])
    fBm <- cumsum(fGn)
    fBm[1] <- 0      ##
    ## ---------------
    ## plot of fBm
    ## ---------------
    if(plotfBm) {
      par(mfrow = c(1, 1))
```

```
        time <- (0:(n - 1))/n
        Nchar <- as.character(n)
        Nleg <- paste(c("N = ", Nchar), collapse = " ")
        Hchar <- as.character(round(H, 3))
        Hleg <- paste(c(", H = ", Hchar), collapse = "")
        NHleg <- paste(c(Nleg, Hleg), collapse = "")
        leg <- paste(c(
          "Path of a fractional Brownian motion ----- parameters",
          NHleg), collapse = " : ")
        plot(time, fBm, type = "l", main = leg)
      }
      fBm
    }
}
#We set starting value of H
Ht <- 0.01

#To store MSE of predictions for all the Hurst parameter values (0.01,0.99)
MSE1 <- c()
MSE2 <- c()
MSE3 <- c()

#To store the accuracy of predictions for all the Hurst parameter values (0.01,0.99)
AC <- c()
AC1 <- c()
AC2 <- c()
AC3 <- c()


while (Ht < 1){
k <- 1

#To store the squared error of predictions for fixed Hurst parameter
SE1 <- c()
SE2 <- c()
SE3 <- c()

#To store whether prediction was correct for fixed Hurst parameter
E1 <- c()
E2 <- c()
E3 <- c()
E <- c()

while (k<= 2000) {

#Generated values of fBm for fixed Hurst parameter H
a <- circFBM(500,Ht,FALSE)
Y <- c()
j<- 0

#Derives the fGn values
while (j <= length(a)-1) {
  Y <- append(Y,a[j+1]-a[j])
  j <- j+1
}

#Tracks the time series in terms of ordinal patterns
Patern_sequence <- c()


#Used to track how many times specific pattern occurs in time series and
#to track how many time specific transition occurs
c1 <- 0
c2 <- 0
c3 <- 0
c4 <- 0
c5 <- 0
c6 <- 0

t11 <- 0
t12 <- 0
t14 <- 0

t26 <- 0
t25 <- 0
t23 <- 0

t31 <- 0
t32 <- 0
t34 <- 0

t45 <- 0
t43 <- 0
t46 <- 0

t51 <- 0
t52 <- 0
t54 <- 0

t66 <- 0
t63 <- 0
t65 <- 0


i <- 0
```

```
#Converting the time series into ordinal patterns
while (i <= length(Y)-3) {
  if (Y[i+1] > 0 & Y[i+2] > 0){
    Patern_sequence <- append(Patern_sequence,1)
    c1 <- c1 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t11 <- t11 + 1
      }
      else if (Patern_sequence[i] == 3){
        t31 <- t31 + 1
      }
      else if (Patern_sequence[i] == 5){
        t51 <- t51 + 1
      }
    }
  }
  else if (Y[i+1]>0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) > 0 ){
    Patern_sequence <- append(Patern_sequence,2)
    c2 <- c2 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t12 <- t12 + 1
      }
      else if (Patern_sequence[i] == 3){
        t32 <- t32 + 1
      }
      else if (Patern_sequence[i] == 5){
        t52 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] > 0 & (Y[i+1]+Y[i+2]) > 0 ){
    Patern_sequence <- append(Patern_sequence,3)
    c3 <- c3 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t23 <- t23 + 1
      }
      else if (Patern_sequence[i] == 4){
        t43 <- t43 + 1
      }
      else if (Patern_sequence[i] == 6){
        t63 <- t63 + 1
      }
    }
  }
  else if (Y[i+1]>0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,4)
    c4 <- c4 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t14 <- t14 + 1
      }
      else if (Patern_sequence[i] == 3){
        t34 <- t34 + 1
      }
      else if (Patern_sequence[i] == 5){
        t54 <- t54 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] > 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,5)
    c5 <- c5 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t25 <- t12 + 1
      }
      else if (Patern_sequence[i] == 4){
        t45 <- t32 + 1
      }
      else if (Patern_sequence[i] == 6){
        t65 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] < 0){
    Patern_sequence <- append(Patern_sequence,6)
    c6 <- c6 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t26 <- t26 + 1
      }
      else if (Patern_sequence[i] == 4){
        t46 <- t46 + 1
      }
      else if (Patern_sequence[i] == 6){
        t66 <- t66 + 1
      }
    }
  }
  i <- i+1
}
```

```
#Predicting using ordinal patterns and noting whether the prediction was correct
if (Patern_sequence[length(Patern_sequence)] == 1){
  if (t11 > t12+t14){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 2){
  if (t23+t25 > t26){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 3){
  if (t31 > t32+t34){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 4){
  if (t43+t45 > t46){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 5){
  if (t51 > t52+t54){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 6){
  if (t63+t65 > t66){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
i<-j-1
H <- Ht

#Calculating the best linear predictor (1 observation) and resulting squared
#error and whether a prediction was correct
Y_est_1 <- (2**(2*H-1) - 1)*Y[i-1]
SE1 <- append(SE1,(Y[i]-Y_est_1)**2)
E1 <- append(E1,abs(sign(Y[i])-sign(Y_est_1)))

#Calculating the best linear predictor (2 observation) and resulting squared
#error and whether a prediction was correct
Y_est_2 <- (2 **(2*H-1) - 1)*(1-(3**(2*H)+1)/2 + 2**(2*H))/(1-(2**(2*H-1)-1)**2)*Y[i-1]
+((3**(2*H)+1)/2 - 2**(2*H) - (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))*Y[i-2]
SE2 <- append(SE2,(Y[i]-Y_est_2)**2)
E2 <- append(E2,abs(sign(Y[i])-sign(Y_est_2)))

#Calculating the best linear predictor (3 observation) and resulting squared
#error and whether a prediction was correct
Y_est_3 <- ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)*((3**(2*H)+1)/2 - 2**(2*H))
*((3**(2*H)+1)/2 - 2**(2*H) - 1)+ (2**(4*H-1)+2**(2*H-1)-3**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))*Y[i-1] + (((2**(2*H-1)-1)**2)*((3**(2*H)+1)/2-2**(2*H)-1)
+((3**(2*H)+1)/2 - 2**(2*H))*(1-((3**(2*H)+1)/2 - 2**(2*H))**2)+(2**(2*H-1)-1)*(2**(4*H-1)
+2**(2*H-1)-3**(2*H))*((3**(2*H)+1)/2-2**(2*H)-1))/(1-(2**(2*H-1)-1)**2 +((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))*Y[i-2] + ((2**(2*H-1)-1)*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)*((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2
- 2**(2*H)-1)+(2**(4*H-1)+2**(2*H-1)-3**(2*H))*(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2
- ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H))) +((3**(2*H)+1)/2
- 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*Y[i-3]
SE3 <- append(SE3,(Y[i]-Y_est_3)**2)
E3 <- append(E3,abs(sign(Y[i])-sign(Y_est_3)))
k <- k+1
}
#Calculating the MSE of each linear predictor for fixed Hurst parameter H
MSE1 <- append(MSE1,mean(SE1))
MSE2 <- append(MSE2,mean(SE2))
MSE3 <- append(MSE3,mean(SE3))

#Calculating the accuracy of all predictions methods for fixed Hurst parameter H
AC <- append(AC,(1-sum(E)/2/length(E)))
AC1 <- append(AC1,1-sum(E1)/2/(k-1))
AC2 <- append(AC2,1-sum(E2)/2/(k-1))
AC3 <- append(AC3,1-sum(E3)/2/(k-1))
```

```
Ht <- Ht + 0.01
}

#Creating a data frame that contains all the MSE
df1 <- data.frame(Method=rep(c("1 previous observation","2 previous observations"
,"3 previous observations"),each=99),H=rep(c(seq(0.01,0.99, by=0.01)),3),MSE = c(MSE1,MSE2,MSE3))

#Creating a plot of this Data frame
MSEplot <- ggplot(df1, aes(x=H, y=MSE, group=Method, linetype = Method)) +
  geom_point(aes(color=Method),size = 3) +
  theme_classic() +
  theme(
    legend.position = c(0.8, 0.8), # Change the coordinates as needed
    legend.background = element_rect(fill = "white", color = "black"),
    legend.box.background = element_rect(color = "black"),
    legend.box = "vertical",  # Make the legend vertical
    axis.title.x = element_text(size = 16), # Change the size of x-axis label
    axis.title.y = element_text(size = 16)  # Change the size of y-axis label
  ) +
  labs(
    x = "H",
    y = "MSE",
    color = "Method",
    linetype = "Method"
  )
# Print the plot
print(MSEplot)

#Creating a data frame that contains all the Accuracy's of predictions
df2 <- data.frame(Method=rep(c("1 previous observation","2 previous observations"
,"3 previous observations","Ordinal pattern"),each=99),H=rep(c(seq(0.01,0.99, by=0.01)),4)
,AC = c(AC1,AC2,AC3,AC))

#Creating a plot of this Data frame
ACplot <- ggplot(df2, aes(x=H, y=AC, group=Method, linetype = Method)) +
  geom_point(aes(color=Method),size = 3) +
  theme_classic() +
  theme(
    legend.position = c(0.9, 0.23), # Change the coordinates as needed
    legend.background = element_rect(fill = "white", color = "black"),
    legend.box.background = element_rect(color = "black"),
    legend.box = "vertical",  # Make the legend vertical
    axis.title.x = element_text(size = 16), # Change the size of x-axis label
    axis.title.y = element_text(size = 16)  # Change the size of y-axis label
  ) +
  labs(
    x = "H",
    y = "Accuracy",
    color = "Method",
    linetype = "Method"
  )
# Print the plot
print(ACplot)
```

B.3 Script 3

This script was used to calculate the error between the real Hurst parameter H and Hurst estimator $\hat{H}$ in section 6.1.

```
library(longmemo)
library(pracma)
library(ggplot2)

circFBM<- function(n, H, plotfBm=FALSE){
  if(missing(n)) n <- 500
  if(missing(H))
    H <- 0.6
  if(missing(plotfBm)) plotfBm <- 1
  ## ------------------------------------------------------------
  ## first line of the circulant matrix, C, built via covariances of fGn
  ## ------------------------------------------------------------
  lineC <- function(n, H, m){
    k <- 0:(m - 1)
    H2 <- 2 * H
    v <- (abs((k - 1)/n)^H2 - 2 * (k/n)^H2 + ((k + 1)/n)^H2)/2
    ind <- c(0:(m/2 - 1), m/2, (m/2 - 1):1)
    v <- v[ind + 1]
    drop(v)
  }
  ##
  ## ---------------------
  ## next power of two > n
  ## ---------------------
  m <- 2
  repeat {
    m <- 2 * m
    if(m >= (n - 1))
      break
  }
  stockm <- m      ##
  ## ------------------------------------------------------
  ## research of the power of two (<2^18) such that
  ## C is definite positive
  ## ------------------------------------------------------
  repeat {
    m <- 2 * m
    eigenvalC <- lineC(n, H, m)
    if (m<100) print(eigenvalC)
    eigenvalC <- Re(fft(c(eigenvalC), inverse = F))
    if((all(eigenvalC > 0)) | (m > 2^18))
      break
  }
  if(m > 2^18) {
    cat("------> exact method, impossible!!", fill = T)
    cat("------> can't find m such that C is definite positive", fill
        = T)
    break
  }
  else {
    ##
    ## ------------------------------------------------------
    ## simulation of W=(Q)^t Z, where Z leads N(0,I_m)
    ## and  (Q)_{jk} = m^(-1/2) exp(-2i pi jk/m)
    ## ------------------------------------------------------
    ar <- rnorm(m/2 + 1)
    ai <- rnorm(m/2 + 1)
    ar[1] <- sqrt(2) * ar[1]
    ar[(m/2 + 1)] <- sqrt(2) * ar[(m/2 + 1)]
    ai[1] <- 0
    ai[(m/2 + 1)] <- 0
    ar <- c(ar[c(1:(m/2 + 1))], ar[c((m/2):2)])
    aic <-  - ai
    ai <- c(ai[c(1:(m/2 + 1))], aic[c((m/2):2)])
    W <- complex(real = ar, imaginary = ai) ##
    ## ---------------------------
    ## reconstruction of the fGn
    ## ---------------------------
    W <- (sqrt(eigenvalC)) * W
    fGn <- fft(W, inverse = F)
    fGn <- (1/(sqrt(2 * m))) * fGn
    fGn <- Re(fGn[c(1:n)])
    fBm <- cumsum(fGn)
    fBm[1] <- 0      ##
    ## ------------
    ## plot of fBm
    ## ------------
    if(plotfBm) {
      par(mfrow = c(1, 1))
      time <- (0:(n - 1))/n
      Nchar <- as.character(n)
      Nleg <- paste(c("N= ", Nchar), collapse = " ")
      Hchar <- as.character(round(H, 3))
      Hleg <- paste(c(", H=", Hchar), collapse = "")
      NHleg <- paste(c(Nleg, Hleg), collapse = "")
      leg <- paste(c(
```

```
        "Path of a fractional Brownian motion ------ parameters",
        NHleg), collapse = " : ")
      plot(time, fBm, type = "l", main = leg)
    }
    fBm
  }
}
#We set starting value of H
Ht <- 0.01

#Used to store the estimated values of Hurst parameter
hurst_e <- c()
while (Ht < 1){
k <- 1
while (k<= 1) {
#Generated values of fBm for fixed Hurst parameter H
a <- circFBM(20000,Ht,FALSE)
Y <- c()
j<- 0

#Derives the fGn values
while (j <= length(a)-1) {
    Y <- append(Y,a[j+1]-a[j])
    j <- j+1
  }

#Tracks the time series in terms of ordinal patterns
Patern_sequence <- c()

#Counts how many time up-down behaviour patterns occur in time series
count <- 0

#Used to track how many times specific pattern occurs in time series and
#to track how many time specific transition occurs
c1 <- 0
c2 <- 0
c3 <- 0
c4 <- 0
c5 <- 0
c6 <- 0

t11 <- 0
t12 <- 0
t14 <- 0

t26 <- 0
t25 <- 0
t23 <- 0

t31 <- 0
t32 <- 0
t34 <- 0

t45 <- 0
t43 <- 0
t46 <- 0

t51 <- 0
t52 <- 0
t54 <- 0

t66 <- 0
t63 <- 0
t65 <- 0


i <- 0
#Converting the time series into ordinal patterns
while (i <= length(Y)-3) {
  if (Y[i+1] > 0 & Y[i+2] > 0){
    Patern_sequence <- append(Patern_sequence,1)
    c1 <- c1 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t11 <- t11 + 1
      }
      else if (Patern_sequence[i] == 3){
        t31 <- t31 + 1
      }
      else if (Patern_sequence[i] == 5){
        t51 <- t51 + 1
      }
    }
  }
  else if (Y[i+1]>0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) > 0 ){
    Patern_sequence <- append(Patern_sequence,2)
    count <- count + 1
    c2 <- c2 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t12 <- t12 + 1
      }
      else if (Patern_sequence[i] == 3){
        t32 <- t32 + 1
      }
```

```
      else if (Patern_sequence[i] == 5){
        t52 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] > 0 & (Y[i+1]+Y[i+2]) > 0 ){
    Patern_sequence <- append(Patern_sequence,3)
    count <- count + 1
    c3 <- c3 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t23 <- t23 + 1
      }
      else if (Patern_sequence[i] == 4){
        t43 <- t43 + 1
      }
      else if (Patern_sequence[i] == 6){
        t63 <- t63 + 1
      }
    }
  }
  else if (Y[i+1]>0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,4)
    count <- count + 1
    c4 <- c4 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t14 <- t14 + 1
      }
      else if (Patern_sequence[i] == 3){
        t34 <- t34 + 1
      }
      else if (Patern_sequence[i] == 5){
        t54 <- t54 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] > 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,5)
    count <- count + 1
    c5 <- c5 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t25 <- t12 + 1
      }
      else if (Patern_sequence[i] == 4){
        t45 <- t32 + 1
      }
      else if (Patern_sequence[i] == 6){
        t65 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] < 0){
    Patern_sequence <- append(Patern_sequence,6)
    c6 <- c6 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t26 <- t26 + 1
      }
      else if (Patern_sequence[i] == 4){
        t46 <- t46 + 1
      }
      else if (Patern_sequence[i] == 6){
        t66 <- t66 + 1
      }
    }
  }
  i <- i+1
}
k <- k + 1
}
Ht <- Ht+0.01

#We calculate the frequency estimator c_n and then calculate Hurst estimator value
c_e <- count/19997
H_e <- max(0,log(cos(pi*c_e/2),2) + 1)
hurst_e <- append(hurst_e,H_e)
}

#Calculating the error between real Hurst parameter and Hurst estimator
o <- 1
error <- c()
while (o <= 99) {
  error <- append(error,hurst_e[o] - o/100)
  o <- o + 1
}

#Creating a data frame that contains all errors between real Hurst parameter
#and Hurst estimator
df1 <- data.frame(Method=rep(c("Error"),each=99),H=c(seq(0.01,0.99, by=0.01)),Error = error)

#Creating a plot of this Data frame
Errorplot <- ggplot(df1, aes(x=H, y=Error, group=Method, linetype = Method)) +
```

```
  geom_line(aes(color=Method),size = 1) +
  theme_classic() +
  theme(
    axis.title.x = element_text(size = 16), # Change the size of x-axis label
    axis.title.y = element_text(size = 16)  # Change the size of y-axis label
  ) +
  labs(
    x = "H",
    y = "Error"
  )

#Print the plot
Errorplot + theme(legend.position="none") + geom_abline(intercept = 0.02, slope = 0)
+ geom_abline(intercept = -0.02, slope = 0)
```

B.4 Script 4

This script was used to generate the moving average process and to compare the accuracy of predictions in Section 6.2. (The code can be adapted to generate values for MA(1) or MA(2), the listed code is for MA(3)).

```
#We set the starting a value to 0
a <- 0

#Used to store the accuracy of different prediction methods
ACOPMR <- c()
ACLPMR <- c()
ACLPMR2 <- c()
ACLPMR3 <- c()


while(a <= 1){
  k <- 1
  hurst_e <- c()

  #To store whether prediction was correct for fixed Hurst parameter
  E <- c()
  e <- 0
  e2 <- 0
  e3 <- 0
  while (k<= 10000) {
    #Generated values of MA(3) for fixed value a
    x<-arima.sim(model=list(ma=c(a/2,a/4,a/8)),n=105)

    #Tracks the time series in terms of ordinal patterns
    Patern_sequence <- c()

    #Counts how many time up-down behaviour patterns occur in time series
    count <- 0

    #Used to track how many times specific pattern occurs in time series and
    #to track how many time specific transition occurs
    c1 <- 0
    c2 <- 0
    c3 <- 0
    c4 <- 0
    c5 <- 0
    c6 <- 0

    t11 <- 0
    t12 <- 0
    t14 <- 0

    t26 <- 0
    t25 <- 0
    t23 <- 0

    t31 <- 0
    t32 <- 0
    t34 <- 0

    t45 <- 0
    t43 <- 0
    t46 <- 0

    t51 <- 0
    t52 <- 0
    t54 <- 0

    t66 <- 0
    t63 <- 0
    t65 <- 0
    i <- 1

    i <- 0
    #Converting the time series into ordinal patterns
    while (i <= length(x)-2) {
      if (x[i] > 0 & x[i+1] > 0){
        Patern_sequence <- append(Patern_sequence,1)
        c1 <- c1 + 1
        if (i > 1){
          if (Patern_sequence[i-1] == 1){
            t11 <- t11 + 1
          }
          else if (Patern_sequence[i-1] == 3){
            t31 <- t31 + 1
          }
          else if (Patern_sequence[i-1] == 5){
            t51 <- t51 + 1
          }
        }
      }
      else if (x[i] > 0 & x[i+1] < 0 & x[i]+x[i+1] > 0){
        Patern_sequence <- append(Patern_sequence,2)
        count <- count + 1
```

```
      c2 <- c2 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 1){
          t12 <- t12 + 1
        }
        else if (Patern_sequence[i-1] == 3){
          t32 <- t32 + 1
        }
        else if (Patern_sequence[i-1] == 5){
          t52 <- t52 + 1
        }
      }
    }
    else if (x[i] < 0 & x[i+1] > 0 & x[i]+x[i+1] > 0){
      Patern_sequence <- append(Patern_sequence,3)
      count <- count + 1
      c3 <- c3 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t23 <- t23 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t43 <- t43 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t63 <- t63 + 1
        }
      }
    }
    else if (x[i] > 0 & x[i+1] < 0 & x[i]+x[i+1] < 0 ){
      Patern_sequence <- append(Patern_sequence,4)
      count <- count + 1
      c4 <- c4 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 1){
          t14 <- t14 + 1
        }
        else if (Patern_sequence[i-1] == 3){
          t34 <- t34 + 1
        }
        else if (Patern_sequence[i-1] == 5){
          t54 <- t54 + 1
        }
      }
    }
    else if (x[i] < 0  & x[i+1] > 0 & x[i]+x[i+1] < 0){
      Patern_sequence <- append(Patern_sequence,5)
      count <- count + 1
      c5 <- c5 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t25 <- t12 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t45 <- t32 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t65 <- t52 + 1
        }
      }
    }
    else if (x[i] < 0 & x[i+1] < 0){
      Patern_sequence <- append(Patern_sequence,6)
      c6 <- c6 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t26 <- t26 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t46 <- t46 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t66 <- t66 + 1
        }
      }
    }
    i <- i+1
  }
  #We calculate the frequency estimator c_n and then calculate Hurst estimator value
  c_e <- count/length(Patern_sequence)
  H_e <- max(0,log(cos(pi*c_e/2),2) + 1)
  hurst_e <- append(hurst_e,H_e)
  H <- H_e

  #Calculating the best linear predictor (1 observation)
  #and whether a prediction was correct
  X_est_1 <- (2**(2*H-1) - 1)*x[length(x)-1]
  if (X_est_1 < 0){
    e <- e + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_1 > 0) {
    e <- e + abs(sign(x[length(x)]) - 1)
  }
```

```
  #Calculating the best linear predictor (2 observation)
  #and whether a prediction was correct
  X_est_2 <- (2 **(2*H-1) - 1)*(1-(3**(2*H)+1)/2 + 2**(2*H))/(1-(2**(2*H-1)-1)**2)
  *x[length(x)-1] + ((3**(2*H)+1)/2 - 2**(2*H)
  - (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))*x[length(x)-2]
  if (X_est_2 < 0){
    e2 <- e2 + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_2 > 0) {
    e2 <- e2 + abs(sign(x[length(x)]) - 1)
  }

  #Calculating the best linear predictor (3 observation)
  #and whether a prediction was correct
  X_est_3 <- ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)*
  ((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H) - 1)
  + (2**(4*H-1)+2**(2*H-1)-3**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2
  - 2**(2*H))))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2
  - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2
  - 2**(2*H))))*x[length(x)-1] + (((2**(2*H-1)-1)**2)*((3**(2*H)+1)/2-2**(2*H)-1)
  +((3**(2*H)+1)/2 - 2**(2*H))*(1-((3**(2*H)+1)/2 - 2**(2*H))**2)+(2**(2*H-1)-1)
  *(2**(4*H-1)+2**(2*H-1)-3**(2*H))*((3**(2*H)+1)/2-2**(2*H)-1))/(1-(2**(2*H-1)-1)**2
  +((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))
  *((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*x[length(x)-2]  + ((2**(2*H-1)-1)
  *((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)
  *((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H)-1)
  +(2**(4*H-1)+2**(2*H-1)-3**(2*H))*(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2
  - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2
  - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*x[length(x)-3]
  if (X_est_3 < 0){
    e3 <- e3 + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_3 > 0) {
    e3 <- e3 + abs(sign(x[length(x)]) - 1)
  }

  #Predicting using ordinal patterns and noting whether the prediction was correct
  if (Patern_sequence[length(Patern_sequence)] == 1){
    if (t11 > t12+t14){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 2){
    if (t23+t25 > t26){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 3){
    if (t31 > t32+t34){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 4){
    if (t43+t45 > t46){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 5){
    if (t51 > t52+t54){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 6){
    if (t63+t65 > t66){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  k <- k + 1
}

#Calculating accuracy of predictions for fixed value a
ACOP <- 1-sum(E)/2/length(E)
ACOPMR <- append(ACOPMR,ACOP)
ACLP <- 1-e/2/(k-1)
ACLPMR <- append(ACLPMR,ACLP)
ACLP2 <- 1-e2/2/(k-1)
ACLPMR2 <- append(ACLPMR2,ACLP2)
```

```
    ACLP3 <- 1-e3/2/(k-1)
    ACLPMR3 <- append(ACLPMR3, ACLP3)
    a = a + 0.1
}

#Prints all the accuracy's of all a in (0,1) with increments of 0.1
print(ACOPMR)
print(ACLPMR)
print(ACLPMR2)
print(ACLPMR3)
```

B.5 Script 5

This script was used to generate autoregressive process and to compare the accuracy of predictions in Section 6.3. (The code can be adapted to generate values for AR(1) or AR(2), the listed code is for AR(3)).

```
#We set the starting a value to 0
a <- 0

#Used to store the accuracy of different prediction methods
ACOPMR <- c()
ACLPMR <- c()
ACLPMR2 <- c()
ACLPMR3 <- c()


while(a <= 1){
  k <- 1
  hurst_e <- c()

  #To store whether prediction was correct for fixed Hurst parameter
  E <- c()
  e <- 0
  e2 <- 0
  e3 <- 0
  while (k<= 10000) {
    #Generated values of AR(3) for fixed value a
    x<-arima.sim(model=list(ar=c(a/2,a/4,a/8)),n=105)

    #Tracks the time series in terms of ordinal patterns
    Patern_sequence <- c()

    #Counts how many time up-down behaviour patterns occur in time series
    count <- 0

    #Used to track how many times specific pattern occurs in time series and
    #to track how many time specific transition occurs
    c1 <- 0
    c2 <- 0
    c3 <- 0
    c4 <- 0
    c5 <- 0
    c6 <- 0

    t11 <- 0
    t12 <- 0
    t14 <- 0

    t26 <- 0
    t25 <- 0
    t23 <- 0

    t31 <- 0
    t32 <- 0
    t34 <- 0

    t45 <- 0
    t43 <- 0
    t46 <- 0

    t51 <- 0
    t52 <- 0
    t54 <- 0

    t66 <- 0
    t63 <- 0
    t65 <- 0
    i <- 1

    i <- 0
    #Converting the time series into ordinal patterns
    while (i <= length(x)-2) {
      if (x[i] > 0 & x[i+1] > 0){
        Patern_sequence <- append(Patern_sequence,1)
        c1 <- c1 + 1
        if (i > 1){
          if (Patern_sequence[i-1] == 1){
            t11 <- t11 + 1
          }
          else if (Patern_sequence[i-1] == 3){
            t31 <- t31 + 1
          }
          else if (Patern_sequence[i-1] == 5){
            t51 <- t51 + 1
          }
        }
      }
      else if (x[i] > 0 & x[i+1] < 0 & x[i]+x[i+1] > 0){
        Patern_sequence <- append(Patern_sequence,2)
        count <- count + 1
```

```
      c2 <- c2 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 1){
          t12 <- t12 + 1
        }
        else if (Patern_sequence[i-1] == 3){
          t32 <- t32 + 1
        }
        else if (Patern_sequence[i-1] == 5){
          t52 <- t52 + 1
        }
      }
    }
    else if (x[i] < 0 & x[i+1] > 0 & x[i]+x[i+1] > 0){
      Patern_sequence <- append(Patern_sequence,3)
      count <- count + 1
      c3 <- c3 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t23 <- t23 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t43 <- t43 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t63 <- t63 + 1
        }
      }
    }
    else if (x[i] > 0 & x[i+1] < 0 & x[i]+x[i+1] < 0 ){
      Patern_sequence <- append(Patern_sequence,4)
      count <- count + 1
      c4 <- c4 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 1){
          t14 <- t14 + 1
        }
        else if (Patern_sequence[i-1] == 3){
          t34 <- t34 + 1
        }
        else if (Patern_sequence[i-1] == 5){
          t54 <- t54 + 1
        }
      }
    }
    else if (x[i] < 0  & x[i+1] > 0 & x[i]+x[i+1] < 0){
      Patern_sequence <- append(Patern_sequence,5)
      count <- count + 1
      c5 <- c5 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t25 <- t12 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t45 <- t32 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t65 <- t52 + 1
        }
      }
    }
    else if (x[i] < 0 & x[i+1] < 0){
      Patern_sequence <- append(Patern_sequence,6)
      c6 <- c6 + 1
      if (i > 1){
        if (Patern_sequence[i-1] == 2){
          t26 <- t26 + 1
        }
        else if (Patern_sequence[i-1] == 4){
          t46 <- t46 + 1
        }
        else if (Patern_sequence[i-1] == 6){
          t66 <- t66 + 1
        }
      }
    }
    i <- i+1
  }
  #We calculate the frequency estimator c_n and then calculate Hurst estimator value
  c_e <- count/length(Patern_sequence)
  H_e <- max(0,log(cos(pi*c_e/2),2) + 1)
  hurst_e <- append(hurst_e,H_e)
  H <- H_e

  #Calculating the best linear predictor (1 observation)
  #and whether a prediction was correct
  X_est_1 <- (2**(2*H-1) - 1)*x[length(x)-1]
  if (X_est_1 < 0){
    e <- e + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_1 > 0) {
    e <- e + abs(sign(x[length(x)]) - 1)
  }
```

```
  #Calculating the best linear predictor (2 observation)
  #and whether a prediction was correct
  X_est_2 <- (2 **(2*H-1) - 1)*(1-(3**(2*H)+1)/2 + 2**(2*H))/(1-(2**(2*H-1)-1)**2)
  *x[length(x)-1] + ((3**(2*H)+1)/2 - 2**(2*H)
  - (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))*x[length(x)-2]
  if (X_est_2 < 0){
    e2 <- e2 + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_2 > 0) {
    e2 <- e2 + abs(sign(x[length(x)]) - 1)
  }

  #Calculating the best linear predictor (3 observation)
  #and whether a prediction was correct
  X_est_3 <- ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)*
  ((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H) - 1)
  + (2**(4*H-1)+2**(2*H-1)-3**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2
  - 2**(2*H))))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2
  - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2
  - 2**(2*H))))*x[length(x)-1] + (((2**(2*H-1)-1)**2)*((3**(2*H)+1)/2-2**(2*H)-1)
  +((3**(2*H)+1)/2 - 2**(2*H))*(1-((3**(2*H)+1)/2 - 2**(2*H))**2)+(2**(2*H-1)-1)
  *(2**(4*H-1)+2**(2*H-1)-3**(2*H))*((3**(2*H)+1)/2-2**(2*H)-1))/(1-(2**(2*H-1)-1)**2
  +((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))
  *((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*x[length(x)-2]  + ((2**(2*H-1)-1)
  *((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)
  *((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H)-1)
  +(2**(4*H-1)+2**(2*H-1)-3**(2*H))*(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2
  - ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2
  - 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*x[length(x)-3]
  if (X_est_3 < 0){
    e3 <- e3 + abs(sign(x[length(x)]) + 1)
  }
  else if (X_est_3 > 0) {
    e3 <- e3 + abs(sign(x[length(x)]) - 1)
  }

  #Predicting using ordinal patterns and noting whether the prediction was correct
  if (Patern_sequence[length(Patern_sequence)] == 1){
    if (t11 > t12+t14){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 2){
    if (t23+t25 > t26){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 3){
    if (t31 > t32+t34){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 4){
    if (t43+t45 > t46){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 5){
    if (t51 > t52+t54){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 6){
    if (t63+t65 > t66){
      E <- append(E,abs(sign(x[length(x)])-1))
    }
    else {
      E <- append(E,abs(sign(x[length(x)])+1))
    }
  }
  k <- k + 1
}

#Calculating accuracy of predictions for fixed value a
ACOP <- 1-sum(E)/2/length(E)
ACOPMR <- append(ACOPMR,ACOP)
ACLP <- 1-e/2/(k-1)
ACLPMR <- append(ACLPMR,ACLP)
ACLP2 <- 1-e2/2/(k-1)
ACLPMR2 <- append(ACLPMR2,ACLP2)
```

```
    ACLP3 <- 1-e3/2/(k-1)
    ACLPMR3 <- append(ACLPMR3, ACLP3)
    a = a + 0.1
}

#Prints all the accuracy's of all a in (0,1) with increments of 0.1
print(ACOPMR)
print(ACLPMR)
print(ACLPMR2)
print(ACLPMR3)
```

B.6 Script 6

This script was used to analyse the Oil prices data in Section 7.1 and make predictions using all discussed prediction methods.

```
library(readxl)
library(tseries)
library(healthyR.ts)
DCOILWTICO <- read_excel("DCOILWTICO.xls")
i <- 11
x <- c()

#Used to convert data into time series
while (i<= 10031){
  x <- append(x,as.numeric(DCOILWTICO$...2[i]))
  i <- i + 1
}
j <- 1
#Used to fill in the missing data points
while (j <= 10021) {
  if (x[j] <= 0.5) {
    index = j + 1
    while (x[index] <= 0.5){
      index = index +1
    }
    diference = index - j
    a = (diference/(diference+1))*x[j-1] + 1/(diference+1)*x[index]
    x[j] <- a
  }
  j <- j+1
}
#Used to plot the time series
function1 <- function(i){x[i]}
curve(function1,from=1, to = 10020, xlab="Day", ylab= "Dollars per Barrel", col=2)

#Used to calculate logarithmic returns R_t
R_t <- c()
k <- 2
while(k <= 10021){
  R_t <- append(R_t, log(x[k])-log(x[k-1]))
  k <- k+1
}
mean(R_t)

#Used to calculate autocovariance for set lag and plot these results
acf_stationary_1 <- acf(R_t, lag.max = length(R_t), plot = FALSE)
plot(acf_stationary_1, ylab = "Autocovariance", col = 2)

#Used to plot the logarithmic returns as time series
function2 <- function(i){R_t[i]}
curve(function2,from=1, to = 10020, xlab="Day", ylab= "Logarithmic return", col=2)

move <- 0
hurst_e <- c()

#To store whether prediction was correct
e1 <- 0
e2 <- 0
e3 <- 0
E <- c()
while(move <= 9918){
  set <- 1 + move
  Y <- c()
  while(set <= 102 + move){
    Y <- append(Y,R_t[set])
    set <- set + 1
  }
  ##Tracks the time series in terms of ordinal patterns
  Patern_sequence <- c()

  #Counts how many time up-down behaviour patterns occur in time series
  count <- 0

  #Used to track how many times specific pattern occurs in time series and
  #to track how many time specific transition occurs
  c1 <- 0
  c2 <- 0
  c3 <- 0
  c4 <- 0
  c5 <- 0
  c6 <- 0

  t11 <- 0
  t12 <- 0
  t14 <- 0

  t26 <- 0
  t25 <- 0
  t23 <- 0

  t31 <- 0
```

```
t32 <- 0
t34 <- 0

t45 <- 0
t43 <- 0
t46 <- 0

t51 <- 0
t52 <- 0
t54 <- 0

t66 <- 0
t63 <- 0
t65 <- 0


i <- 0
#Converting the time series into ordinal patterns
while (i <= length(Y)-3) {
  if (Y[i+1] >= 0 & Y[i+2] >= 0){
    Patern_sequence <- append(Patern_sequence,1)
    c1 <- c1 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t11 <- t11 + 1
      }
      else if (Patern_sequence[i] == 3){
        t31 <- t31 + 1
      }
      else if (Patern_sequence[i] == 5){
        t51 <- t51 + 1
      }
    }
  }
  else if (Y[i+1]>= 0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) >= 0 ){
    Patern_sequence <- append(Patern_sequence,2)
    count <- count + 1
    c2 <- c2 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t12 <- t12 + 1
      }
      else if (Patern_sequence[i] == 3){
        t32 <- t32 + 1
      }
      else if (Patern_sequence[i] == 5){
        t52 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] >= 0 & (Y[i+1]+Y[i+2]) >= 0 ){
    Patern_sequence <- append(Patern_sequence,3)
    count <- count + 1
    c3 <- c3 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t23 <- t23 + 1
      }
      else if (Patern_sequence[i] == 4){
        t43 <- t43 + 1
      }
      else if (Patern_sequence[i] == 6){
        t63 <- t63 + 1
      }
    }
  }
  else if (Y[i+1]>= 0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,4)
    count <- count + 1
    c4 <- c4 + 1
    if (i > 0){
      if (Patern_sequence[i] == 1){
        t14 <- t14 + 1
      }
      else if (Patern_sequence[i] == 3){
        t34 <- t34 + 1
      }
      else if (Patern_sequence[i] == 5){
        t54 <- t54 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] >= 0 & (Y[i+1]+Y[i+2]) < 0 ){
    Patern_sequence <- append(Patern_sequence,5)
    count <- count + 1
    c5 <- c5 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t25 <- t12 + 1
      }
      else if (Patern_sequence[i] == 4){
        t45 <- t32 + 1
      }
      else if (Patern_sequence[i] == 6){
```

```
        t65 <- t52 + 1
      }
    }
  }
  else if (Y[i+1]<0 & Y[i+2] < 0){
    Patern_sequence <- append(Patern_sequence,6)
    c6 <- c6 + 1
    if (i > 0){
      if (Patern_sequence[i] == 2){
        t26 <- t26 + 1
      }
      else if (Patern_sequence[i] == 4){
        t46 <- t46 + 1
      }
      else if (Patern_sequence[i] == 6){
        t66 <- t66 + 1
      }
    }
  }
  i <- i+1
}
c_e <- count/length(Patern_sequence)
H_e <- max(0,log(cos(pi*c_e/2),2) + 1)
hurst_e <- append(hurst_e,H_e)
H <- H_e

#Calculating the best linear predictor (1 observation)
#and checking whether a prediction was correct
X_est_1 <- (2**(2*H-1) - 1)*R_t[set-2]
if (X_est_1 < 0){
  e1 <- e1 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_1 > 0) {
  e1 <- e1 + abs(sign(Y[length(Y)]) - 1)
}

#Calculating the best linear predictor (2 observation)
#and checking whether a prediction was correct
X_est_2 <- (2 **(2*H-1) - 1)*(1-(3**(2*H)+1)/2 + 2**(2*H))/(1-(2**(2*H-1)-1)**2)
*R_t[set-2] + ((3**(2*H)+1)/2 - 2**(2*H)
- (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))*R_t[set-3]
if (X_est_2 < 0){
  e2 <- e2 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_2 > 0) {
  e2 <- e2 + abs(sign(Y[length(Y)]) - 1)
}

#Calculating the best linear predictor (3 observation)
#and checking whether a prediction was correct
X_est_3 <- ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)
*((3**(2*H)+1)/2 - 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H) - 1)
+ (2**(4*H-1)+2**(2*H-1)-3**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))*R_t[set-2] + (((2**(2*H-1)-1)**2)
*((3**(2*H)+1)/2-2**(2*H)-1)+((3**(2*H)+1)/2 - 2**(2*H))*(1-((3**(2*H)+1)/2
- 2**(2*H))**2)+(2**(2*H-1)-1)*(2**(4*H-1)+2**(2*H-1)-3**(2*H))*((3**(2*H)+1)
/2-2**(2*H)-1))/(1-(2**(2*H-1)-1)**2 +((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2
- 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))*R_t[set-3]  + ((2**(2*H-1)-1)*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)*((3**(2*H)+1)/2 - 2**(2*H))
*((3**(2*H)+1)/2 - 2**(2*H)-1)+(2**(4*H-1)+2**(2*H-1)-3**(2*H))
*(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))
*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))*R_t[set-4]
if (X_est_3 < 0){
  e3 <- e3 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_3 > 0) {
  e3 <- e3 + abs(sign(Y[length(Y)]) - 1)
}

#Predicting using ordinal patterns and noting whether the prediction was correct
if (Patern_sequence[length(Patern_sequence)] == 1){
  if (t11 > t12+t14){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 2){
  if (t23+t25 > t26){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 3){
  if (t31 > t32+t34){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
```

```
    else {
      E <- append(E,abs(sign(Y[length(Y)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 4){
    if (t43+t45 > t46){
      E <- append(E,abs(sign(Y[length(Y)])-1))
    }
    else {
      E <- append(E,abs(sign(Y[length(Y)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 5){
    if (t51 > t52+t54){
      E <- append(E,abs(sign(Y[length(Y)])-1))
    }
    else {
      E <- append(E,abs(sign(Y[length(Y)])+1))
    }
  }
  if (Patern_sequence[length(Patern_sequence)] == 6){
    if (t63+t65 > t66){
      E <- append(E,abs(sign(Y[length(Y)])-1))
    }
    else {
      E <- append(E,abs(sign(Y[length(Y)])+1))
    }
  }
  move <- move + 1
}

#Calculating the accuracy of all predictions methods.
ACOP <- 1-sum(E)/2/length(E)
ACLP1 <- 1-e1/2/(move)
ACLP2 <- 1-e2/2/(move)
ACLP3 <- 1-e3/2/(move)

#Print all the accuracy's
print(ACOP)
print(ACLP1)
print(ACLP2)
print(ACLP3)
```

B.7 Script 7

This script was used to analyse the global temperature data in Section 7.2 and make predictions using all discussed prediction methods.

```
library(readxl)
library(forecast)
library(healthyR.ts)
library(tseries)
#Used to read the Global temperature data and covert into time series and plot
GWT <- read_excel("C:/Users/hjaci/Desktop/R codes for BA/GWT_partial.xlsx")
ts.data1 = ts(data=as.vector(t(GWT['Temperature'])), start = c(1900),end = c(2016),
              frequency = 12)
plot(ts.data1, xlab="Year", ylab="Average temperature", col = 2)

#Decomposes the time series and plots the resulting components
decompose = decompose(ts.data1)
plot(decompose, xlab="Year", col = 2)

#Removes the seasonal and trend components of time series
deseason=seasadj(decompose)
Random = diff(deseason)
X <- as.numeric(Random)

#Used to calculate autocovariance for set lag and plot these results
acf_stationary_1 <- acf(X, lag.max = length(X), plot = FALSE)
plot(acf_stationary_1, ylab="Autocovariance", col = 2)

move <- 0
hurst_e <- c()

#To store whether prediction was correct
e1 <- 0
e2 <- 0
e3 <- 0
E <- c()
mean(X)
while(move <= 1272){
  set <- 1 + move
  Y <- c()
  while(set <= 120 + move){
    Y <- append(Y,X[set])
    set <- set + 1
  }

  #Tracks the time series in terms of ordinal patterns
  Patern_sequence <- c()

  #Counts how many time up-down behaviour patterns occur in time series
  count <- 0

  #Used to track how many times specific pattern occurs in time series and
  #to track how many time specific transition occurs
  c1 <- 0
  c2 <- 0
  c3 <- 0
  c4 <- 0
  c5 <- 0
  c6 <- 0

  t11 <- 0
  t12 <- 0
  t14 <- 0

  t26 <- 0
  t25 <- 0
  t23 <- 0

  t31 <- 0
  t32 <- 0
  t34 <- 0

  t45 <- 0
  t43 <- 0
  t46 <- 0

  t51 <- 0
  t52 <- 0
  t54 <- 0

  t66 <- 0
  t63 <- 0
  t65 <- 0


  i <- 0
  #Converting the time series into ordinal patterns
  while (i <= length(Y)-3) {
    if (Y[i+1] >= 0 & Y[i+2] >= 0){
      Patern_sequence <- append(Patern_sequence,1)
      c1 <- c1 + 1
```

```
      if (i > 0){
        if (Patern_sequence[i] == 1){
          t11 <- t11 + 1
        }
        else if (Patern_sequence[i] == 3){
          t31 <- t31 + 1
        }
        else if (Patern_sequence[i] == 5){
          t51 <- t51 + 1
        }
      }
    }
    else if (Y[i+1]>= 0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) >= 0 ){
      Patern_sequence <- append(Patern_sequence,2)
      count <- count + 1
      c2 <- c2 + 1
      if (i > 0){
        if (Patern_sequence[i] == 1){
          t12 <- t12 + 1
        }
        else if (Patern_sequence[i] == 3){
          t32 <- t32 + 1
        }
        else if (Patern_sequence[i] == 5){
          t52 <- t52 + 1
        }
      }
    }
    else if (Y[i+1]<0 & Y[i+2] >= 0 & (Y[i+1]+Y[i+2]) >= 0 ){
      Patern_sequence <- append(Patern_sequence,3)
      count <- count + 1
      c3 <- c3 + 1
      if (i > 0){
        if (Patern_sequence[i] == 2){
          t23 <- t23 + 1
        }
        else if (Patern_sequence[i] == 4){
          t43 <- t43 + 1
        }
        else if (Patern_sequence[i] == 6){
          t63 <- t63 + 1
        }
      }
    }
    else if (Y[i+1]>= 0 & Y[i+2] < 0 & (Y[i+1]+Y[i+2]) < 0 ){
      Patern_sequence <- append(Patern_sequence,4)
      count <- count + 1
      c4 <- c4 + 1
      if (i > 0){
        if (Patern_sequence[i] == 1){
          t14 <- t14 + 1
        }
        else if (Patern_sequence[i] == 3){
          t34 <- t34 + 1
        }
        else if (Patern_sequence[i] == 5){
          t54 <- t54 + 1
        }
      }
    }
    else if (Y[i+1]<0 & Y[i+2] >= 0 & (Y[i+1]+Y[i+2]) < 0 ){
      Patern_sequence <- append(Patern_sequence,5)
      count <- count + 1
      c5 <- c5 + 1
      if (i > 0){
        if (Patern_sequence[i] == 2){
          t25 <- t12 + 1
        }
        else if (Patern_sequence[i] == 4){
          t45 <- t32 + 1
        }
        else if (Patern_sequence[i] == 6){
          t65 <- t52 + 1
        }
      }
    }
    else if (Y[i+1]<0 & Y[i+2] < 0){
      Patern_sequence <- append(Patern_sequence,6)
      c6 <- c6 + 1
      if (i > 0){
        if (Patern_sequence[i] == 2){
          t26 <- t26 + 1
        }
        else if (Patern_sequence[i] == 4){
          t46 <- t46 + 1
        }
        else if (Patern_sequence[i] == 6){
          t66 <- t66 + 1
        }
      }
    }
    i <- i+1
  }
  c_e <- count/length(Patern_sequence)
```

```
H_e <- max(0,log(cos(pi*c_e/2),2) + 1)
hurst_e <- append(hurst_e,H_e)
H <- H_e

#Calculating the best linear predictor (1 observation)
#and checking whether a prediction was correct
X_est_1 <- (2**(2*H-1) - 1)*X[set-2]
if (X_est_1 < 0){
  e1 <- e1 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_1 > 0) {
  e1 <- e1 + abs(sign(Y[length(Y)]) - 1)
}

#Calculating the best linear predictor (2 observation)
#and checking whether a prediction was correct
X_est_2 <- (2 **(2*H-1) - 1)*(1-(3**(2*H)+1)/2 + 2**(2*H))/(1-(2**(2*H-1)-1)**2)
*X[set-2] + ((3**(2*H)+1)/2 - 2**(2*H)
- (2**(2*H - 1 )-1)**2)/(2 ** (2*H) - 2**(4*H-2))*X[set-3]
if (X_est_2 < 0){
  e2 <- e2 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_2 > 0) {
  e2 <- e2 + abs(sign(Y[length(Y)]) - 1)
}

#Calculating the best linear predictor (3 observation)
#and checking whether a prediction was correct
X_est_3 <- ((2**(2*H-1)-1)*(1-(2**(2*H-1)-1)**2)+(2**(2*H-1)-1)*((3**(2*H)+1)/2
- 2**(2*H))*((3**(2*H)+1)/2 - 2**(2*H) - 1)+ (2**(4*H-1)+2**(2*H-1)-3**(2*H))
*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H))))/(1-(2**(2*H-1)-1)**2
- ((2**(2*H-1)-1)**2)*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2
- 2**(2*H))*((2**(2*H-1)-1)**2 - ((3**(2*H)+1)/2 - 2**(2*H)))*X[set-2]
+ (((2**(2*H-1)-1)**2)*((3**(2*H)+1)/2-2**(2*H)-1)+((3**(2*H)+1)/2 - 2**(2*H))
*(1-((3**(2*H)+1)/2 - 2**(2*H))**2)+(2**(2*H-1)-1)*(2**(4*H-1)+2**(2*H-1)-3**(2*H))
*((3**(2*H)+1)/2-2**(2*H)-1))/(1-(2**(2*H-1)-1)**2 +((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H)))*X[set-3]  + ((2**(2*H-1)-1)*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H)))+(2**(2*H-1)-1)*((3**(2*H)+1)/2 - 2**(2*H))
*((3**(2*H)+1)/2 - 2**(2*H)-1)+(2**(4*H-1)+2**(2*H-1)-3**(2*H))
*(1-(2**(2*H-1)-1)**2))/(1-(2**(2*H-1)-1)**2 - ((2**(2*H-1)-1)**2)
*(1-((3**(2*H)+1)/2 - 2**(2*H)))+((3**(2*H)+1)/2 - 2**(2*H))*((2**(2*H-1)-1)**2
- ((3**(2*H)+1)/2 - 2**(2*H))))*X[set-4]
if (X_est_3 < 0){
  e3 <- e3 + abs(sign(Y[length(Y)]) + 1)
}
else if (X_est_3 > 0) {
  e3 <- e3 + abs(sign(Y[length(Y)]) - 1)
}
if (Patern_sequence[length(Patern_sequence)] == 1){
  if (t11 > t12+t14){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}

#Predicting using ordinal patterns and noting whether the prediction was correct
if (Patern_sequence[length(Patern_sequence)] == 2){
  if (t23+t25 > t26){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 3){
  if (t31 > t32+t34){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 4){
  if (t43+t45 > t46){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 5){
  if (t51 > t52+t54){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
  else {
    E <- append(E,abs(sign(Y[length(Y)])+1))
  }
}
if (Patern_sequence[length(Patern_sequence)] == 6){
  if (t63+t65 > t66){
    E <- append(E,abs(sign(Y[length(Y)])-1))
  }
```

```
    else {
      E <- append(E,abs(sign(Y[length(Y)])+1))
    }
  }
  move <- move + 1
}

#Calculating the accuracy of all predictions methods.
ACOP <- 1-sum(E)/2/length(E)
ACLP1 <- 1-e1/2/(move)
ACLP2 <- 1-e2/2/(move)
ACLP3 <- 1-e3/2/(move)

#Print all the accuracy's
print(ACOP)
print(ACLP1)
print(ACLP2)
print(ACLP3)
```