7CCSMRPJ
Directional Change Intrinsinc Time Framework as Target Transformation in Time Series Modelling

Final Project Report

Yu-Kuan, Lin

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

List of Figures **6**

List of Tables **7**

1 Introduction **9**

1.1 Aims and Objectives 9
1.2 Problem Statement 10
1.3 Movination 10
1.4 Report Structure 10

2 Background **11**

2.1 Univariate Time Series Forecasting 11
2.1.1 General Notions of Time Series Forecasting 11
2.1.2 Machine Learning Regression Modelling 15
2.2 Target Transformation in Univariate Time Series Forecasting 21
2.3 Directional Change Intrinsic Time Framework 23

3 Literature Review **24**

3.1 Target Transformation 24
3.2 Directional Change intrinsic time framework 28

4 Methodology **31**

5 Results and Evaluation **32**

6 Conclusion **33**

Bibliography **34**

A chapter one **39**

A.1 ch1 sec 1 39

B chapter two **40**

B.1 ch2 sec 1 40

B.2 ch2 sec 2 40

C chapter three **41**

C.1 ch3 sec 1 41

List of Figures

1.1 ETS's prediction on the test set 10

List of Tables

List of Algorithms

1 Validation 19

Chapter 1

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

1.1 Aims and Objectives

something about chapter one, section 1

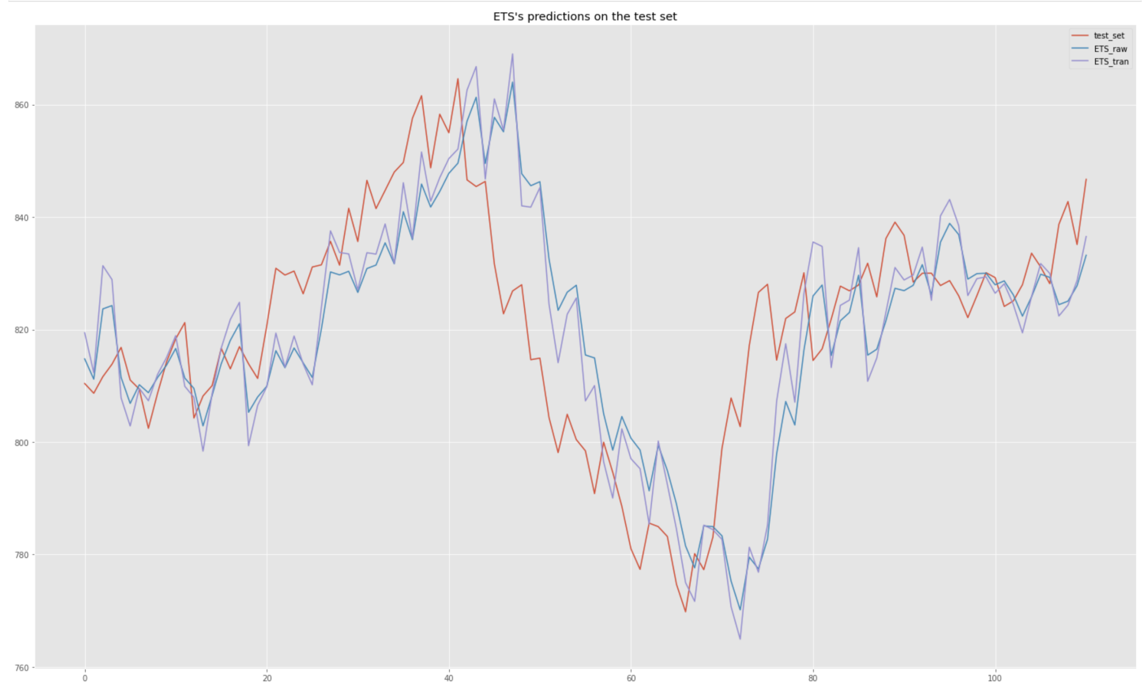

Figure 1.1: ETS's prediction on the test set

1.2 Problem Statement

something

1.3 Movination

something

1.4 Report Structure

something

Chapter 2

Background

This chapter presents a comprehensive theoretical background to the topics related to our experiment and analyses.

2.1 Univariate Time Series Forecasting

In this section, we discuss the topic of univariate time series forecasting.

2.1.1 General Notions of Time Series Forecasting

A time series is a sequential collection of random variables indexed by time. Suppose the random variables are of one dimension. The time series is considered univariate[1]. In this section, the discussions fall within the context of univariate time series forecasting. Let $Y = \{Y_{t_i}\}_{i=1,2,\ldots,n}$ be an univariate time series with $Y_{t_i} \in \mathbb{R},\ \forall i,\ n \in \mathbb{N}$ and $t = \{t_i\}_{i=1,2,\ldots,n}$ being the set of time indices of Y (also referred to as the set of timestamps). Let $\mathcal{Y}_{t_k}$ be the largest information set about Y that is accessible to the model at time point $t_k \in t$, e.g., we might have the observations of Y being known ($\{Y_{t_j} = y_{t_j}\}_{j=1,2,\ldots,k} \subset \mathcal{Y}_{t_k}$). Then in the context of modelling, for an unknown (random) target Y_{t_s} with $s > k,\ s \in \mathbb{N}$, we can articulate the notion of forecasting as the following:

> Forecasting the value Y_{t_s} at time point t_k is to find a function $f(\mathcal{Y}_{t_k}) = \widehat{Y_{t_s}}$, such that $\widehat{Y_{t_s}}$ is a good estimation of Y_{t_s}.

[1]If the random variables are of dimension higher than one, then the time series is considered multivariate. Datasets in such form are also referred to as panel data.

We can develop all sorts of functions f for such forecasting objectives. Functions devised to serve the objective are referred to as models. In a general sense, the notion of modelling refers to the methodologies of devising a model that serves the objective well. In particular, for any arbitrary pair of t_k and t_s, we want to have a model $f(\mathcal{Y}_{t_k})$ such that it gives us a reliable estimation of Y_{t_s}.

Gap

Notice that a modelling objective is parameterised by the pair of time indices (t_k, t_s). The timestamp t_k directly affects the information set $\mathcal{Y}_{t_k}$ and thus determines the information the model f can utilise. The timestamp t_s, on the other hand, controls how far in the future we are forecasting. If the gap between the two timestamps is big, the model is asked to forecast further into the future. If the gap between the two timestamps is small, then the objective might be considered easier because we are only trying to look a tiny step ahead into the future. In order to better communicate and characterise the forecasting objective, we formalise this gap between the pair of timestamps as the *gap* and denote it as τ:

$$\tau = t_s - t_k$$

With such a notion of the gap, we can then articulate the forecasting objective as τ ahead forecasting. τ takes the format of the timestamps. Depending on the format of the timestamps, the objective can be one-step ahead forecasting or one-year ahead forecasting. We will go into topics concerning timestamps in one of the upcoming paragraphs.

Forecast Horizon

Observe that the target we have in the previous forecasting objective Y_{t_s} is a single value in the future. It is possible to generalise the target and have multiple targets in the future. The number of targets we try to forecast is called the *forecast horizon*. Having a forecast horizon equal to one is to forecast one value into the future, and having a forecast horizon equal to five is to forecast all five values into the future. To put it formally, let $\langle \cdot \rangle$ be a counting operation that counts the number of elements for a finite set, and let a task have a finite collection of unknown values $Y_S = \{Y_{s_1}, Y_{s_2}, \cdots\}_{s_1, s_2, \ldots \in t,\ s_1, s_2, \ldots > t_k}$. Then the forecast horizon of such a task is denoted as

$$H = \langle Y_S \rangle$$

Timestamps

In a time series, the time index of a random variable carries information about the time point in which the random variable lives in the time domain. In some sense, the timestamps mark the 'location' of the random variables on a timeline. For example, a monthly revenue time series Y in an arbitrary year can have the set of months in a year as its timestamp set and be denoted as $Y = \{Y_{Jan}, Y_{Feb}, Y_{Mar}, \cdots, Y_{Dec}\}$. The time indices also tell us the time-relevance (geological relationship on a timeline) of the random variables among each other. In fact, the time-relevance of the random variables in a time series plays a crucial role in time series analysis. We often have to perform mathematical operations involving such relationships. An example is our coming up with the gap measurement we addressed in the previous paragraph. Another example is the calculation of the relative growth of the time series. The need for these math operations pushes modellers to devise innovative ways to define the timestamps because we cannot easily perform calculations on notations like *September* or $Friday$. To put it in math terms, what we often do is to have a mapping from physical timestamps to the real number line (or a subset of the real number set, say, the natural number set) and use the target set of this mapping as the timestamp set for math operations. In the next paragraph, we discuss some examples of such mapping.

Take the previous monthly revenue time series as an example; one simplest way is to index the time series chronologically with natural numbers $\{1, 2, 3, \cdots, 12\}$. The new index system allows for mathematical operations on the timestamps, such as addition. The objective of two-month ahead forecasting can be considered as two-step ahead forecasting with $\tau = 2$. Three-month moving average of the time series can now be of a generalised form of a three-step moving average. Another good example is the financial studies of stochastic processes, in which we often use the non-negative real line and adopt an annual scale, i.e., the starting point of the time series is indexed as 0, one month after that is indexed 0.0833, one-year time point is indexed 1.0, and so on. This is particularly useful when we expand the time series studies using Stochastic Differential Equations (SDE). Let the SDE of a stochastic process dY_t be given as

$$\frac{dY_t}{Y_t} = \mu_t dt + \sigma_t dW_t, \quad dW_t \sim N(0, dt).$$

dt in the drift term is now properly defined as a real number on which we can do all sorts of math operations (observe how dW_t is defined as a Brownian Motion that follows a Gaussian distribution with variance dt).

Time Heterogeneity

There are cases where the timestamps of the time series are not identically spaced between consecutive random variables. The previous example of monthly revenue in a year is one of them due to the months having different durations. Time series as such is technically referred to as being *time heterogeneous*[2]. Time heterogeneity can be an interesting source of information carried by the time series but can also be a major issue in time series studies. The following paragraph presents a common problem caused by time heterogeneity.

Calculating measurements related to the unit of time can be a problem with time heterogeneous time series. One common example of such measurement is the return used in finance. Return measures the relative change of the price (or, say, value) over a period of time with respect to its initial level. Several definitions can be drawn to the notion of return, but we will look at the simplest one as they all exhibit the same relationship with time heterogeneity. Let $Y = \{Y_{t_i}\}_{i=1,2,\ldots n},\ n \in \mathbb{N}$ now be a time series of the price movement of an asset over time, and the time index set being $t = \{t_i\}_{i=1,2,\ldots,n}$. Define the corresponding net return measure as

$$R_{t_i} = \frac{Y_{t_i} - Y_{t_{i-1}}}{Y_{t_{i-1}}}, \quad i \in \{2, 3, \cdots, n\}. \tag{2.1}$$

The net return R_{t_i} for some i is thus the relative price change of Y_{t_i} over the time period $t_i - t_{i-1}$ with respect to $Y_{t_{i-1}}$. Let $R = \{R_{t_i}\}_{i=2,3,\ldots,n}$ be the time series of one-step net returns derived from Y. If t is equally spaced by, say, a day, the time series Y is time-homogeneous. The time series R we calculated is a time series of daily net return of Y. Nevertheless, in the case where t is not equally spaced, the time series Y is time heterogeneous. Then we no longer know the period of the net returns R we calculated from the time series Y. This is an example of how time heterogeneity can complicate time series analyses. Time heterogeneity in finance has been studied a lot, especially with the popularisation of electronic systems. See Dacarogna et al. (2001) for more information.

[2]Time heterogeneity is common in financial time series due to the nature of how financial markets work. For example, most markets are open only during working hours on working days. Another example is the high-frequency financial time series (see Dacarogna et al. 2001). *Time homogeneity* is the counterpart of time heterogeneity, specifying time series which have equally spaced timestamps

2.1.2 Machine Learning Regression Modelling

In this section, we further discuss univariate time series modelling and address how we approach the articulated forecasting objective with Machine Learning (ML) regression modelling. In addition to addressing the approach, we also provide relevant statistical notions that can be seen as the underlying theoretical foundation of the standard machine learning procedure.

Recall the forecasting objective is to come up with a model f capable of generating 'good' estimations of some target Y_{t_s} at time t_k using the provided information $\mathcal{Y}_{t_k}$. We will see in the later paragraphs that this is very similar to the process of solving for a Quasi-Maximum Likelihood Estimation (QMLE) in a statistical sense (White (1982)). In the context of modelling, the procedure is normally to gather the available information and formulate it into an optimisation problem: we define a fitness measure (like the 'L' in QMLE), which should be a function of the estimates $f(\mathcal{Y}_{t_k})$ and the target, and we then optimise the fitness as an objective function with respect to the model f in some algorithmic way (the 'M' in QMLE). The final output of such a procedure will be a model (function f) that optimally serves our objective (the 'E' in QMLE). In the remainder of this section, we discuss the general framework of how the procedure works.

The Sliding Window, Design Matrix and Target

In regression problems, the design matrix and the target are made from accessible information and formulated for a modelling environment. In this paragraph, we discuss how to develop the design matrix and target in univariate time series modelling. Recall that the forecasting objective is to estimate Y_{t_s} at time point t_k, with the gap $\tau = t_s - t_k > 0$ and the model is only able to utilise the accessible information set $\mathcal{Y}_{t_k}$. This dataset is called the *training set*. For our modelling problem, we can only use what is provided in our training set $\mathcal{Y}_{t_k}$. Without loss of generality, for the target Y_{t_s}, the information set accessible is $\mathcal{Y}_{t_s-\tau}$. In the event we have a one dimensional time series, $\mathcal{Y}_{t_s-\tau}$ is simply the collection of realised values of Y until time point $t_s - \tau = t_k$, namely

$$\mathcal{Y}_{t_s-\tau} = \{Y_{t_s-\tau} = y_{t_s-\tau}, Y_{t_{s-1}-\tau} = y_{t_{s-1}-\tau}, Y_{t_{s-2}-\tau} = y_{t_{s-2}-\tau}, \cdots, Y_{t_1} = y_{t_1}\}.$$

To make use of this long list of past observations, the idea is to create a sandbox in which we simulate the model making predictions. The environment in which the model makes a prediction is simply a mapping from the information it can use to a target. We do this by using the methodology called the *sliding window*. The sliding window is parameterised by a single constant parameter $\lambda,\ \lambda \in \mathbb{N},\ \lambda > 2$ that controls the width of the window. The decision of λ should take into account modelling configurations including the size of our training set, forecast horizon, gap, and information we want the model to use in a single prediction task. Once λ is decided, we move the window chronologically throughout $\mathcal{Y}_{t_s-\tau}$. For every step of the window, we create an independent prediction instance for the model using the values contain within the window - the final value in a single window is the target for horizon one forecasting and the rest of the values are potentially accessible to the model for its prediction. We then have numerous such forecasting instances for the model from which we can foster the whole sandbox. The sandbox consists of the design matrix $\mathbf{X}$ and the target $\mathbf{y}$. They can be formulated as

$$\mathbf{y} = \begin{bmatrix} y_{t_\lambda} \\ y_{t_{\lambda+1}} \\ \cdot \\ \cdot \\ y_{t_{k-1}} \\ y_{t_k} \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} y_{t_\lambda-\tau} & y_{t_{\lambda-1}-\tau} & \cdots & y_{t_1} \\ y_{t_{\lambda+1}-\tau} & y_{t_\lambda-\tau} & \cdots & y_{t_2} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y_{t_{k-1}-\tau} & y_{t_{k-2}-\tau} & \cdots & y_{t_{k-\lambda}-\tau} \\ y_{t_k-\tau} & y_{t_{k-1}-\tau} & \cdots & y_{t_{k-\lambda+1}-\tau} \end{bmatrix}.$$

The target $\mathbf{y}$ is a $k-\lambda+1$ by 1 column matrix and the design matrix $\mathbf{X}$ is of $k-\lambda+1$ by λ. In this training environment, the model has $k-\lambda+1$ predictions to make, each being to use a row vector in $\mathbf{X}$, denoted as $\mathbf{X}_i,\ i \in \{1, 2, \cdots, k-\lambda+1\}$ and estimate the corresponding row element in $\mathbf{y}$, denoted as $\mathbf{y}_i$. The idea is to find a model that best maps the rows in $\mathbf{X}$ to elements in $\mathbf{y}$ in general, and we say this is our best model f that serves the objective of a τ ahead forecasting task given the time series we have.

The Number of Lags

One remark regarding the design matrix is the notion of the *number of lags*. The number of lags describes how many latest observations the model is allowed to use for a single forecasting task, i.e., the number of columns in $\mathbf{X}$. Its naming originates in autoregressive estimation in time series studies. Such estimation aims at finding the optimal order of the autoregressive feature of a time series. This is equivalent to finding the optimal number of latest observations the model should use for a single prediction. Typically, such autoregressive order is referred to as the *lag*.

Modelling

We start by describing what a model is in more detail. Given the design matrix $\mathbf{X}$ and target $\mathbf{y}$ we made in the previous section, the forecasting objective is now transformed into coming up with a model f that best maps the rows in $\mathbf{X}$ to the corresponding element in $\mathbf{y}$.

Consider f as an arbitrary machine learning regression model with a known structure. Knowing the structure of f implies we know the structure of its parameter set and how f maps $\mathbf{X}_i$ to $\mathbf{y}_i$ for some i. Let θ be the mapping that generates the parameters that go into f. The output of θ is parameterised by its input set; let it be ϕ. Parameter set like ϕ is called the *hyperparameters* - it characterises the parameters of f. The model f in our forecasting objective can thus be noted as

$$f(\theta(\phi); \mathbf{X}) = \widehat{\mathbf{y}} \sim \mathbf{y}.$$

To better measure the performance of such an estimation, we define a fitness function $\mathcal{E}$. The fitness function takes the estimations generated by f and returns a real number signifying how well they fit the target. We can finally formulate our modelling objective as

$$\arg_{\theta(\phi)} \max \mathcal{E}(\widehat{\mathbf{y}}, \mathbf{y}), \quad \widehat{\mathbf{y}} = f(\theta(\phi); \mathbf{X}). \tag{2.2}$$

This process is what we call *model training*.

To give an example, if f is a simple linear regression model, then we know θ gives a tuple of real numbers (known as weights) which the model uses to generate a linear combination of its inputs; in this case, a row matrix $\mathbf{X}_i$. Specifications of θ are then controlled by its input ϕ, e.g., whether there is an intercept term or the number of elements of the tuple[3]. Not knowing the exact values of θ (and certainly ϕ as well) describes the state of the model f being untrained. Then the modelling objective is to find ϕ and θ such that $\mathcal{E}(f(\theta(\phi); \mathbf{X}), \mathbf{y})$ is maximised.

The Statistical Resemblance

Analogously, training a machine learning model can be put into statistical terms. In particular, it resembles the Quasi-Maximum Likelihood Estimation (QMLE) process with some minor tweaks. Consider our objective with Y, but with the elements following an arbitrary distribution $\mathcal{D}$ characterised by θ, i.e., $Y_{t_i} \sim \mathcal{D}(\theta), \ \forall \ t_i \in$

[3]Note that the number of elements in the tuple depends on the number of lags we have in making the design matrix $\mathbf{X}$, i.e., $\lambda \in \phi$.

t^4. Consider an artibrary pair of timestamps (t_k, t_s) and let $\mathcal{F}_{Y_{t_s}|\mathcal{Y}_{t_k}}(\cdot;\theta)$ be the conditional joint probability density function (pdf) of the random variable Y_{t_s} given $\mathcal{Y}_{t_k}$ and θ. Then the expression

$$\mathcal{F}_{Y_{t_s}|\mathcal{Y}_{t_k}}(y_{t_s}|\mathcal{Y}_{t_k};\theta)$$

describes the probability of observing $Y_{t_s} = y_{t_s}$ conditional on the past observations and parameter θ. Let the value of $Y_{t_s} = y_{t_s}$ be given, then the objective of QMLE is to find the θ conditional on observing $\mathcal{Y}_{t_k}$, under which y_{t_s} is most likely to be observed. We can formalise such objectives as

$$\arg_\theta \max \mathcal{F}_{Y_{t_s}|\mathcal{Y}_{t_k}}(\theta; y_{t_s}|\mathcal{Y}_{t_k}).$$

Notice how the variable is now θ while the observations are given. The new objective function to be optimised is called the *likelihood function*, denoted as $\mathcal{L}(\theta)$. $\mathcal{L}(\theta)$ is essentially still a pdf. It returns the probability of observing y_{t_s} conditional on $\mathcal{Y}_{t_k}$ with the parameter θ. Maximising such likelihood function with respect to θ is thus equivalent to finding a θ dictating the generating mechanism of the random variable Y_{t_s} such that it is most likely to be observed as y_{t_s}.

Training a machine learning model bears some resemblance to QMLE. Training a model aims to find a mechanism to reproduce the target as the observation. At the same time, QMLE tries to find the set of parameters characterising the distribution of the target variable conditional on its past realisations. Both methodologies tackle the problem with an optimisation framework involving an objective function: QMLE utilises the probability density function (pdf) that comes with a random variable with a known distribution. ML modelling devises a fitness function to evaluate the goodness of the estimation. The outcome of training an ML model is to have a deterministic function that generates the target and serves the purpose of forecasting. On the other hand, QMLE yields a set of parameters dictating the underlying distribution of the target variables, i.e., you end up having a recipe to construct a probabilistic distribution not deterministicly generate a target value. We hope this section contributes to a better theoretical understanding of ML modelling in terms of statistical analysis.

[4] The 'Quasi-' simply means we do not know whether the distribution $\mathcal{D}$ is Gaussian or not (see White 1982).

Model Selection with Validation

In the phase of training a model using $\mathcal{Y}_{t_k}$, notice that we have to first have the hyperparameters ϕ before we actually starting searching for optimal θ. Such process of finding ϕ is called *model selection* or *hyperparameter tuning*. The way we approach model selection starts by selecting a subset of the training set with respect to the row and call it the *validation set*[5]. Let the size of the validation set be v, $v \in \mathbb{N}$, $v \ll k - \lambda + 1$[6]. We then try (validate) whether a ϕ candidate works fine using the validation set. For a given hyperparameter candidate ϕ_0, we know the exact structure of $\theta(\phi_0)$ such that we can run the optimisation regime, i.e., model training using equation 2.2. Then for ϕ_0, we do the *validation* illustrated as algorithm 1.

Algorithm 1 Validation

Let L be an empty list.
Let $s = k - \lambda + 1 - v$ be the first index of the validation set.
for $i \in \{0, 1, 2, \cdots, v-1\}$ **do**
  $\mathbf{y}_{train} \leftarrow \{\mathbf{y}_i\}_{i=1,2,\ldots,s-1+i}$ Use information before the validation point.
  $\mathbf{X}_{train} \leftarrow \{\mathbf{X}_i\}_{i=1,2,\ldots,s-1+i}$ Use information before the validation point.
  Do training: $\theta(\phi_0)_0 \leftarrow \arg_{\theta(\phi_0)} \max \mathcal{E}(\widehat{\mathbf{y}_{train}}, \mathbf{y}_{train}),\ \widehat{\mathbf{y}_{train}} = f(\theta(\phi_0); \mathbf{X}_{train})$
  $\mathbf{y}_{val} \leftarrow \mathbf{y}_{s+i}$ This is the validation instance in $\mathbf{y}$.
  $\mathbf{X}_{val} \leftarrow \mathbf{X}_{s+i}$ This is the validation instance in $\mathbf{X}$.
  Compute fitness score for the $\theta(\phi_0)_0$: $\nu \leftarrow \mathcal{E}(f(\theta(\phi_0)_0; \mathbf{X}_{val}), \mathbf{y}_{val})$
  Store the fitness score ν in list L
end for
Let $V_{\phi_0} \equiv \frac{1}{v} \sum L$ be the mean of the scores stored in list L.

The result for validating a single hyperparameter ϕ_0 is the validation score V_{ϕ_0} we compute in the final step of the algorithm. Normally, we will come up with a set of k candidate hyperparameters $\theta = \{\theta_i\}_{i=1,2,\ldots,k}$, $k \in \mathbb{N}$ and repeat such validation process k times and yield k validation scores $\{V_{\phi_i}\}$, $\forall i \in \{1, 2, \cdots, k\}$. We then take the hyperparameter set with the most preferable fitness score

$$\phi_{best} = \arg_{\phi_i} \max\{V_{\phi_i}\}_{i=1,2,\ldots,k}$$

and say ϕ_{best} is the optimal hyperparameter set to use in our modelling problem.

[5]The choice of validation set is usually a proportion of the latest instances from the training set, say, the latest ten instances or the latest ten percents of instances of the training set.

[6]The decision of v depends on the size of our training dataset. The bigger v we wish to have, the less data is left for the training.

The set of k candidate hyperparameters ϕ is referred to as the *hyperparameter space*. And hyperparameter tuning is the act of searching for an optimal hyperparameter in the hyperparameter space.

Model Evaluation - Testing

In the event we wish to conclude the model's performance using some measure, this is called *model evaluation* or *testing the model*. Testing is done in a similar manner as the validation we addressed in model selection (notice how we use a validation score to measure the performance of a model). In some cases, we could simply use the validation score of the best performing hyperparameter $V_{\phi_{best}}$ because this score is also a concluding measure of our model performance. However, if we want to actually simulate the situation in which our model is put into production, then we can simply take another segment from the training data (a proportion of latest observations) and perform a single round of validation, except for this time, it is not validation anymore. We refer this to as model evalutaion and the dataset used for it as the *test set*. Let κ denote the size of the test dataset. Then we have to push the validation set and let the latest κ instances in the training set be the test set.

Our formation of the sandbox is now turned and renamed into

$$\mathbf{y}_{train} = \begin{bmatrix} y_{t_\lambda} \\ y_{t_{\lambda+1}} \\ \cdot \\ \cdot \\ y_{t_{k-1-v-\kappa}} \\ y_{t_{k-v-\kappa}} \end{bmatrix}, \quad \mathbf{X}_{train} = \begin{bmatrix} y_{t_{\lambda-\tau}} & y_{t_{\lambda-1-\tau}} & \cdots & y_{t_1} \\ y_{t_{\lambda+1-\tau}} & y_{t_{\lambda-\tau}} & \cdots & y_{t_2} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y_{t_{k-1-v-\kappa-\tau}} & y_{t_{k-2-v-\kappa-\tau}} & \cdots & y_{t_{k-\lambda-v-\kappa-\tau}} \\ y_{t_{k-v-\kappa-\tau}} & y_{t_{k-1-v-\kappa-\tau}} & \cdots & y_{t_{k-\lambda+1-v-\kappa-\tau}} \end{bmatrix}$$

$$\mathbf{y}_{val} = \begin{bmatrix} y_{t_{k-v-\kappa+1}} \\ y_{t_{k-v-\kappa+2}} \\ \cdot \\ \cdot \\ y_{t_{k-v-1}} \\ y_{t_{k-v}} \end{bmatrix}, \quad \mathbf{X}_{val} = \begin{bmatrix} y_{t_{k-v-\kappa+1-\tau}} & y_{t_{k-v-\kappa-\tau}} & \cdots & y_{t_{k-\lambda+1-v-\kappa+1-\tau}} \\ y_{t_{k-v-\kappa+2-\tau}} & y_{t_{k-v-\kappa+1-\tau}} & \cdots & y_{t_{k-\lambda+1-v-\kappa+2-\tau}} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y_{t_{k-v-1-\tau}} & y_{t_{k-v-2-\tau}} & \cdots & y_{t_{k-\lambda+1-v-1-\tau}} \\ y_{t_{k-v-\tau}} & y_{t_{k-v-1-\tau}} & \cdots & y_{t_{k-\lambda+1-v-\tau}} \end{bmatrix}$$

$$\mathbf{y}_{test} = \begin{bmatrix} y_{t_{k-v+1}} \\ y_{t_{k-v+2}} \\ \cdot \\ \cdot \\ y_{t_{k-1}} \\ y_{t_k} \end{bmatrix}, \quad \mathbf{X}_{test} = \begin{bmatrix} y_{t_{k-v+1-\tau}} & y_{t_{k-v-\tau}} & \cdots & y_{t_{k-\lambda+1-v+1}} \\ y_{t_{\lambda+1-\tau}} & y_{t_{\lambda-\tau}} & \cdots & y_{t_{k-\lambda+1-v+2}} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y_{t_{k-1-\tau}} & y_{t_{k-2-\tau}} & \cdots & y_{t_{k-\lambda+1-1-\tau}} \\ y_{t_{k-\tau}} & y_{t_{k-1-\tau}} & \cdots & y_{t_{k-\lambda+1-\tau}} \end{bmatrix}$$

To sum up, the *training set* now consists of $k-\lambda+1-v-\kappa$ instances, the *validation set* consists of v instances and the *test set* consists of κ instances. The three sets take up all the information we have in our accessible information set $\mathcal{Y}_{t_k}$. Given we already have the best performing hyperparameter $V_{\phi_{best}}$, we can simply do a single round of validation described in algorithm 1 using the test set and come up with a score that concludes the model performance. Such score is referred to as the *test score*. And that concludes a typical modelling operation.

2.2 Target Transformation in Univariate Time Series Forecasting

Building on the ML regression modelling we established in the previous section, we can now address the topic of target transformation in univariate time series forecasting. For a forecasting task on a time series $Y = \{Y_{t_i}\}_{i=1,2,\ldots}$, we try to devise a model f that deterministicly produce predictions. We utilise all information available and formulate a modelling procedure goes as the following:

1. Construct the corresponding training set $(\mathbf{y}_{train}, \mathbf{X}_{train})$, the validation set $(\mathbf{y}_{val}, \mathbf{X}_{val})$ and the test set $(\mathbf{y}_{test}, \mathbf{X}_{test})$ based on the objective (forecast horizon, gap), modelling configurations (sliding window size, validation size, test size).

2. To find the optimal hyperparameter ϕ_{best}, come up with a set of hyperparameters $\phi = \{\phi_i\}_{i=1,2,\ldots,k}$ and conduct model selection using the validation procedure and the validation set.

3. Finally, test and report the test score of the model with the optimal hyperparameter ϕ_{best} using the test set.

Observe that aside from our appointing the model f and the hyperparameter set ϕ, the model we obtain through the procedure rely heavily on the information contained within the dataset (training, validation and test) precisely because our formulation of the modelling procedure. In other words, the utility of our model depends heavily on the information set (dataset). This gives rise to a large number of studies trying to find more clever ways to process the information before putting it into the optimisation operation with the goal of boosting model performance. Such additional layer of operation on datasets in the modelling procedure is called *data preprocessing*. Moreover, if the additional operation is performed only on the design matrix $\mathbf{X}$ such operations are referred to as *feature engineering*[7]. When the operation is performed also on the target $\mathbf{y}$, it is called *target transformation* or *response transformation*[8]. In the rest of the section, we cover the technical notions of target transformation.

Let $\mathcal{T} : \mathbb{R}^M \longmapsto \mathbb{R}^M$ be an arbitrary target transformation that maps a M dimensional vector to another M dimensional vector. Throughout the modelling procedure, target transformation is to be embedded before every optimisation stated in equation 2.2. Consider an arbitrary optimisation in the modelling procedure to be conducted with the information set $\mathcal{Y}_{t_k} = \{y_{t_i}\}_{i=1,2,\ldots,k}$, we practice

$$\mathcal{Y}'_{t_k} = \mathcal{T}(\mathcal{Y}_{t_k})$$

and use the transformed time series $\mathcal{Y}'_{t_k}$ for the rest of the optimisation operation: given a hyperparameter ϕ_0, we generate the training target $\mathbf{y}'_{train}$, training design

[7]In regression operations, the columns in the design matrix are referred to as features or independent variables.

[8]In regression operations, the target, being a column matrix, is also called the response variable or dependent variable

matrix $\mathbf{X}'_{train}$ with $\mathcal{Y}'_{t_k}$ and perform the optimisation

$$\theta(\phi_0)_0 = \arg_{\theta(\phi_0)} \max \mathcal{E}(\widehat{\mathbf{y}'_{train}}, \mathbf{y}'_{train}),\ \widehat{\mathbf{y}'_{train}} = f(\theta(\phi_0); \mathbf{X}'_{train}).$$

After the optimisation using the transformed time series, the usual next step is to make the validation target, validation design matrix and compute the fitness score of $\theta(\phi_0)_0$. However, there is an optional operation that might be needed depending on the nature of the transformation $\mathcal{T}$. If $\mathcal{T}$ changes the scale the original time series, i.e., $\mathcal{Y}_{t_k}$ and $\mathcal{Y}'_{t_k}$ have different scale, then a *back transformation* is needed to reverse the scale of the prediction generated by f back to its original level. For example, an usual taret transformation used in financial studies is the *log-difference* transformation. The log-difference operation is a simulation of the return we mentioned in equation 2.1.

2.3 Directional Change Intrinsic Time Framework

Chapter 3

Literature Review

Target transformation techniques have been researched as part of the modelling process over the past sixty years. The same applies to the analyses of the underlying mechanisms in financial dynamics. This paper builds on both research directions and analyses the Directional Change (DC) intrinsic time framework as a target transformation technique in time series forecasting. To the best of our knowledge, as this paper was written, we are unaware of any other attempt to bring these two methodologies together. In this chapter, we cover the most relevant works in both realms. We first review some target transformation developments in Section 3.1 and then look at some of the advancements in the Directional Change intrinsic time framework in Section 3.2.

3.1 Target Transformation

In most cases, time series analyses yielded from modelling are justified conditionally on the assumptions made by the models about the data. If the assumptions are not met by the original data, applying some transformation (often non-linear) to the data can help generate these conditions. This has led to various transformation techniques for these types of purposes. In this section, we go through some important assumptions made by the models and some of the most popular corresponding transformations found in the literature.

Homoscedasticity condition[1] is one of the most common assumptions for models

[1] The homoscedasticity (homogeneity of variances) condition requires the variances of different subsets of the sample to be the same. In the case of time series modelling, it is equivalent to

involving statistical inference. As a reference to the analysis of variance, M. S. Bartlett (1947) provided one of the earliest summaries of transformations on raw data addressing this. He covered parametric transformations used in stabilising the variance of modelling error, especially for Poisson and Binomial distributed variables where the variance is a known function of the mean. He discussed, both theoretically and empirically, some of the optimal scales and families of transformations to choose from given different prerequisites. His work showed that modelling tasks do benefit from suitable transformations.

Another common assumption made by the models is normality. Many statistical inference methods assume the variable of interest is normally distributed, including t-test, Analysis of Variance (ANOVA) and Linear Regression. Therefore, it is of paramount importance to transform the data in a way that such conditions are met, if possible. Box & Cox (1964) made a major contribution in this regard by proposing the well-known Box-Cox transformation. The Box-Cox transformation includes both power and logarithmic transformations. It aims at achieving normality of the observations and has been popular in developing modelling methodologies ever since (see Atkinson et al. 2021 and Osborne 2010). An example of applying Box-Cox and achieving better model performance was given in C. Bergmeier et al. (2016). Combined with the widespread exponential smoothing (ETS) forecasting method and the bootstrap aggregation (bagging) technique in machine learning, they proposed a bagging exponential smoothing method using STL decomposition and Box-Cox transformation. The proposed method was tested on the M3 competition dataset and achieved better results than all the original M3 participants (see Makridakis et al. 2000 for the M3 competition).

The method published by Box & Cox, however, is only valid for positive real values. Modifications of the Box-Cox transformation have thus been proposed to address this constraint. A significant extension was proposed by Bickel & Doksum (1981). They embedded a sign function to the power transformation such that the transformation function covers the whole real line. Nevertheless, this modification has its shortcomings: it was shown by Yeo & Johnson (2000) that Bickel & Doksum's modified version of the transformation handles skewed distribution poorly. Yeo & Johnson pointed out the reason being the signed power transformation was designed primarily for handling kurtosis, thus losing its edge concerning skewness. Following up, Yeo & Johnson proposed a new version of the power transformation in the same

requiring a constant variance throughout time.

publication (2000). Their transformation is a generalised version of the Box-Cox transformation and approximates normality while being well-defined on the real line and inducing appropriate symmetricity.

In the previous paragraphs we have described transformation techniques that can be used to satisfy mathematical and statistical conditions assumed by the models. Meeting these assumptions improves the robustness of the conclusions drawn by the modelling results. On a higher level, one can say that the transformations help the models to get better at 'learning' the problem such that they generate more robust outputs. In the upcoming paragraphs, we take on this perspective of treating the transformation techniques as helpers in terms of the learning process of the models and look at some transformation techniques very different from what was covered previously.

Decomposition methods constitute a significant category of techniques that can be used to transform the target in order to help the models learn from the target variable. These methods decompose the mixture of information contained within the observation into patterns, trends, cycles, or other dynamics that are easier to model, i.e., easier for models to learn from. A good example of such methods includes the large number of studies on Fourier-styled transformations in time series modelling[2] (see Kay & Marple 1981 and Bloomfield 2004). Typical Fourier methodologies build on transforming the observations sampled from the time domain into the frequency domain and decomposing them into more informative signals. Out of the great history and advancements of spectrum analysis, we selectively provide a brief overview of some relevant methodologies in the context of target transformation.

Building on Fourier methods, a novel type of transformations operates in the time-frequency domain, i.e., these methods generate time-frequency representations of the observations. The empirical mode decomposition (EMD) proposed by Huang et al. (1998) is one of such key methodologies. EMD decomposes the original time series into 'intrinsic mode functions' (IMF). The IMF's carry information of the underlying structures contained in the observations and can then be used for modelling tasks. The family of wavelet transform methods constitutes another class of methods that operates in the time-frequency domain (see Daubechies 1992 and Percival & Walden 2000). Shensa et al. 1992 first proposed and provided a framework for the discrete wavelet transform (DWT), which belongs to the wavelet transform family. DWT filters the original time series in several folds and yields

[2]Such studies are also called spectrum analysis.

a denoised version of the observation. The information carried by the transformed time series is then more clear and possibly easier to learn. The rationale of using such techniques as target transformation in time series modelling is to provide the models with a more informative, e.g., less noisy, dataset to learn from. We hope this extra procedure helps with the ultimate goal to produce a better trained (learned) model.

Another important family of decomposition methods consists of statistical-related methods. These decomposition methods generally consider the time series as a mixture of three components: seasonal, trend and remainder. They are often used to filter different information contained within the time series (see Wang, Smith, & Hyndman 2006). For a real-world example, monthly unemployment data are usually presented after removing the seasonality. The resulting time series is hence more indicative of the variation of the general economy instead of seasonal disturbance (see Chapter 3.2 in Hyndman & Athanasopoulos 2021). The STL decomposition proposed by Cleveland et al. 1990 has been a robust method for decomposition into seasonal and trend signals. The abbreviation stands for Seasonal and Trend decomposition using Loess. STL considers a time series as a sum (additive) or product (multiplicative) of the seasonal, trend and remainder components. STL is flexible and applicable to many use cases as it can handle any type of seasonality. Its flexibility also resides in its allowance for the user to have control over the time-varying seasonal component and smoothness of the trend cycle. The X-11 method and the Seasonal Extraction in ARIMA Time Series (SEATS) procedure are time series models that rely heavily on seasonal and trend decompositions. They have had many variants and are favoured by official statistical agencies around the world (see Dagum & Bianconcini 2016)[3]. One of the state-of-the-art variants of this family is the X-13ARIMA-SEATS method produced, distributed and maintained by the US Census Bureau (see US Census Bureau 2012 and Monsell & Blakely 2013). It inherits powerful features from X-11, SEATS, and ARIMA methodologies while specialising in seasonal adjustment in extensive time series modelling. The model is conveniently accessible online[4].

[3] X-11 was initially developed by the US Census Bureau, and SEATS was created by the Bank of Spain.

[4] A webpage demonstration of the model is accessible on `http://www.seasonal.website/`; the open-source implementation of the model can be found in the `seasonal` package in R, and a distributed version can be found in the US Census Bureau website `https://www.census.gov/data/software/x13as.X-13ARIMA-SEATS.html`.

3.2 Directional Change intrinsic time framework

The technical core of the Directional Change (DC) intrinsic time framework is quite simple; it is an algorithm (the DC dissection) that samples a time series and yields a new time series, which is a subset of the original observations. By analysing the properties of the resulting time series, it has been found that despite the simplicity of this algorithm, it provides powerful perspectives for looking at market dynamics in the time domain. In this section, we first look at critical works that contribute to the advancements of the DC intrinsic time framework. Then we cover some applications that further developed the framework's value by trying to harness its potential.

Like the developments of many frameworks, the DC intrinsic time framework started out being simple and has developed over time. Guillaume et al. (1997) first published the Directional Change dissection algorithm. The algorithm was presented and used to generate a set of measurements (statistics, variables), from which the authors presented a set of stylised facts found empirically in the spot intra-daily foreign exchange (FX) markets. These stylised facts shed new light on our understanding of market dynamics, especially concerning micro-structure topics, including time-heterogeneity, price formation, market efficiency, liquidity, and both the modelling and the learning process of the market. A little more than a decade later, Tsang formalised the definition of a Directional Change in Tsang (2010), and Glattfelder et al. (2011) discovered a set of twelve scaling laws derived from the DC sampling algorithm. The discovery of the laws added a theoretical foundation to the DC dissection algorithm as it was shown that the output time series carries not only qualitative information (stylised facts) but also interesting quantitative properties. As the DC dissection algorithm's ability to extract information has been studied, it has given rise to the methodology becoming a framework (see Tsang's introduction of a set of profiles (indicators) derived under the DC framework in Tsang (2015) and (2017)).

A well-known fact about analysing financial time series is that the source of many of the challenges can be traced back to the use of physical time (see Dacarogna et al. (2001)). Aloud et al. (2012) discussed the potential of studying financial time series using the DC framework (referred to as the DC approach in their paper) resides in its underlying 'intrinsic time' paradigm. They pointed out that mapping financial time series from the physical time to event-based intrinsic time is the key to how the approach filters out irrelevant information and disturbance observed in the dataset and generates valuable market insights of our interests. Inspired by the studies of complex systems, Petrov et al. (2018) took a different route of demonstrating this point with the use of agent-based modelling. They created a market with trading agents that operate in event-based intrinsic time and found that the price movements generated under such conditions experience statistical properties we observed in real-world physical time FX markets. Such reproduction of real-world stylised facts is another indication of the intrinsic time mechanism being one of the contributing factors to the market dynamics. Recently, Glattfelder & Golub (2022) derived an analytical relationship between physical and intrinsic time based on the scaling laws. In particular, the expression they derived decomposes the movements of the physical-time time series into volatility and liquidity components expressed in intrinsic time. That allows us to explicitly characterise the dynamics observed in physical time using its intrinsic-time representation.

As DC intrinsic time framework becomes theoretically sound, applications building on the framework have been devised. Golub et al. (2016) introduced the Intrinsic Network - an event-based framework based on directional changes. Combining the Intrinsic Network and information theory, they devised a liquidity measure that was shown to be able to predict market stress in terms of liquidity shocks. In Golub et al. (2018), the liquidity measure was integrated with other implementations derived from the DC framework and an algorithmic trading strategy called The Alpha Engine was introduced. The Alpha Engine has several interesting features. First, the bare-bones version of the model (without tweaking) has been shown to be robust, profitable, and can be implemented in real-time. Second, Alpha Engine provides liquidity in the market, i.e., it opens long positions when other market players intend to short and vice versa. The Alpha Engine thus contributes to the healthiness of the market as a participant. Third, Alpha Engine 'beats' random walk processes - it is shown to be profitable even on price dynamics generated by a random walk. Within the context of volatility and risk management in finance, Petrov et al. (2019a) proposed an instantaneous volatility measure under the DC intrinsic time framework. They found seasonality patterns and long memory of volatility through empirical studies. Their work contributes not only to the development of practical tools but also to the understanding of the underlying stochastic drive of financial dynamics. Two further generalisations of the DC framework have been proposed. First, Petrov et al. (2019b) brought the framework into multidimensional space by extending the analytical expressions yielded from one-dimensional analyses to multidimensional space. Their methodology implies that previous works in one-dimensional space (analytical insights, empirical findings, and all the tools and implementations) can be extended to higher dimensions. Another generalisation was developed with respect to the types of stochastic processes (Mayerhofer (2019)). These generalisations, as well as the advancements of the DC intrinsic time framework discussed previously, are indicative of the framework's promising potential worthy of further exploration.

In this chapter, we reviewed relevant works in two different realms: target transformations being used as an additional layer in modelling and the DC intrinsic time framework being a rising methodology. The literature surveyed demonstrates excellent potential in both research directions that justifies our curiosity in bringing them together. In the next chapter, we go into detail about the methodologies of combing the framework and target transformation in time series modelling.

Chapter 4

Methodology

Chapter 5

Results and Evaluation

Chapter 6

Conclusion

Bibliography

A. Adegboye and M. Kampouridis. Machine learning classification and regression models for predicting directional changes trend reversal in fx markets. *Expert Systems with Applications*, 173:114645, 2021.

S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah. Time-series clustering - a decade review. *Information Systems*, 53:16–38, 2015. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2015.04.007. URL `https://www.sciencedirect.com/science/article/pii/S0306437915000733`.

N. Ahmed, A. Atiya, N. Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29:594–621, 08 2010. URL `https://www.tandfonline.com/doi/abs/10.1080/07474938.2010.481556`.

M. Aloud, E. Tsang, R. Olsen, and A. Dupuis. A directional-change event approach for studying financial time series. *Economics*, 6(1), 2012.

A. C. Atkinson, M. Riani, and A. Corbellini. The box–cox transformation: Review and extensions. *Statistical Science*, 36(2):239–255, 2021.

M. S. Bartlett. The use of transformations. *Biometrics*, 3(1):39–52, 1947. ISSN 0006341X, 15410420. URL `http://www.jstor.org/stable/3001536`.

C. Bergmeir, R. J. Hyndman, and J. M. Benítez. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International journal of forecasting*, 32(2):303–312, 2016.

P. J. Bickel and K. A. Doksum. An analysis of transformations revisited. *Journal of the american statistical association*, 76(374):296–311, 1981.

P. Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

G. E. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

U. C. Bureau. *X-13ARIMA-SEATS Reference Manual, Version 1.0*. U.S. Census Bureau, U.S. Department of Commerce, 2012.

R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. Stl: A seasonal-trend decomposition. *J. Off. Stat*, 6(1):3–73, 1990.

E. B. Dagum and S. Bianconcini. *Seasonal adjustment methods and real time trend-cycle estimation*. Springer, 2016.

I. Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.01.001. URL `https://www.sciencedirect.com/science/article/pii/S0169207006000021`. Twenty five years of forecasting.

J. D. Farmer and D. Foley. The economy needs agent-based modelling. *Nature*, 460 (7256):685–686, 2009.

W. A. Fuller. *Introduction to statistical time series*. John Wiley & Sons, 2009.

R. Gençay, M. Dacorogna, U. A. Muller, O. Pictet, and R. Olsen. *An introduction to high-frequency finance*. Elsevier, 2001.

J. B. Glattfelder and A. Golub. Bridging the gap: Decoding the intrinsic nature of time in market data. *arXiv preprint arXiv:2204.02682*, 2022.

J. B. Glattfelder, A. Dupuis, and R. B. Olsen. Patterns in high-frequency fx data: discovery of 12 empirical scaling laws. *Quantitative Finance*, 11(4):599–614, 2011.

A. Golub, G. Chliamovitch, A. Dupuis, and B. Chopard. Multi-scale representation of high frequency market liquidity. *Algorithmic Finance*, 5(1-2):3–19, 2016.

A. Golub, J. B. Glattfelder, and R. B. Olsen. The alpha engine: Designing an automated trading algorithm. In *High-Performance Computing in Finance*, pages 49–76. Chapman and Hall/CRC, 2018.

P. Goodwin et al. The holt-winters approach to exponential smoothing: 50 years old and going strong. *Foresight*, 19(19):30–33, 2010.

V. M. Guerrero. Time-series analysis supported by power transformations. *Journal of forecasting*, 12(1):37–48, 1993.

D. M. Guillaume, M. M. Dacorogna, R. R. Davé, U. A. Müller, R. B. Olsen, and O. V. Pictet. From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. *Finance and stochastics*, 1(2): 95–129, 1997.

N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 454(1971):903–995, 1998.

J. C. Hull. *Options futures and other derivatives.* Pearson Education India, 2003.

R. Hyndman and G. Athanasopoulos. *Forecasting: Principle and Practice, 3rd edition.* OTexts: Melbourne, Australia, 2021. URL `OTexts.com/fpp3`. Accessed in June 2022.

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

J. Hämäläinen and T. Kärkkäinen. Problem transformation methods with distance-based learning for multi-target regression. 10 2020.

S. M. Kay and S. L. Marple. Spectrum analysis—a modern perspective. *Proceedings of the IEEE*, 69(11):1380–1419, 1981.

A. J. Koning, P. H. Franses, M. Hibon, and H. Stekler. The m3 competition: Statistical tests of the results. *International Journal of Forecasting*, 21(3): 397–409, 2005. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2004.10.003. URL `https://www.sciencedirect.com/science/article/pii/S0169207004000810`.

S. Makridakis and M. Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL `https://www.sciencedirect.com/science/article/pii/S0169207019301128`. M4 Competition.

E. Mayerhofer. Three essays on stopping. *Risks*, 7(4):105, 2019.

A.-H. Mihov, N. Firoozye, and P. Treleaven. Towards augmented financial intelligence. *Available at SSRN*, 2022.

B. Monsell and C. Blakely. X-13arima-seats and imetrica. *US Census Bureau, Washington, DC*, 2013.

J. Osborne. Improving your data transformations: Applying the box-cox transformation. *Practical Assessment, Research, and Evaluation*, 15(1):12, 2010.

D. B. Percival and A. T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge university press, 2000.

V. Petrov, A. Golub, and R. B. Olsen. Agent-based model in directional-change intrinsic time. *Available at SSRN 3240456*, 2018.

V. Petrov, A. Golub, and R. Olsen. Instantaneous volatility seasonality of high-frequency markets in directional-change intrinsic time. *Journal of Risk and Financial Management*, 12(2):54, 2019a.

V. Petrov, A. Golub, and R. B. Olsen. Intrinsic time directional-change methodology in higher dimensions. *Available at SSRN 3440628*, 2019b.

M. J. Shensa et al. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing*, 40(10):2464–2482, 1992.

E. Tsang. Directional changes, definitions. *Working Paper WP050-10 Centre for Computational Finance and Economic Agents (CCFEA), University of Essex Revised 1, Tech. Rep.*, 2010.

E. P. Tsang, R. Tao, and S. Ma. Profiling financial market dynamics under directional changes. *Quantitative finance, http://www. tandfonline. com/doi/abs/10.1080/14697688.2016*, 1164887, 2015.

E. P. Tsang, R. Tao, A. Serguieva, and S. Ma. Profiling high-frequency equity price movements in directional changes. *Quantitative finance*, 17(2):217–225, 2017.

X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364, 2006.

H. White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the econometric society*, pages 1–25, 1982.

I.-K. Yeo and R. A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 12 2000. ISSN 0006-3444. doi: 10.1093/biomet/87.4.954. URL `https://doi.org/10.1093/biomet/87.4.954`.

Appendix A

chapter one

Something about chapter one

Another something ...

And another something ...

A.1 ch1 sec 1

Something about chapter one, section 1

Appendix B

chapter two

Something about chapter two

B.1 ch2 sec 1

Something about chapter two, section 1

B.2 ch2 sec 2

Something about chapter two, section 2

Appendix C

chapter three

Something about chapter three

C.1 ch3 sec 1