CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

BENCHMARKING NONSTATIONARY TIME SERIES PREDICTION

Rebecca Pontes Salles

Dissertation submitted to the Postgraduate Program in Computer Science of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of master.

Advisor: Eduardo Soares Ogasawara
Co-advisor: Pedro Henrique González Silva

Rio de Janeiro,
February 2019

BENCHMARKING NONSTATIONARY TIME SERIES PREDICTION

Dissertation submitted to the Postgraduate Program in Computer Science of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of master.

Rebecca Pontes Salles

Examining jury:

President, Professor D.Sc. Eduardo Soares Ogasawara (CEFET/RJ) (advisor)

Professor D.Sc. Pedro Henrique González Silva (CEFET/RJ) (co-advisor)

Professor D.Sc. Eduardo Bezerra da Silva (CEFET/RJ)

Professor D.Sc. Fabio Andre Machado Porto (LNCC)

Professor Dr. Florent Masseglia (INRIA)

Rio de Janeiro,

February 2019

CEFET/RJ – Sistema de Bibliotecas / Biblioteca Central

S168 Salles, Rebecca Pontes
Benchmarking nonstationary time series prediction / Rebecca Pontes Salles.—2019.
xv, 110f. : il. , grafs. , tabs. ; enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2019.
Bibliografia : f. 97-110
Orientador : Eduardo Soares Ogasawara
Coorientador : Pedro Henrique Gonzalez Silva

1. Computação. 2. Processamento de dados. 3. Análise de séries temporais. 4. Framework. I. Ogasawara, Eduardo Soares (Orient.). II. Silva, Pedro Henrique Gonzalez (Coorient.). III. Título.

CDD 004

Elaborada pela bibliotecária Mariana Oliveira CRB-7/5929

DEDICATION

To God and to my beloved family who have always helped, supported and guided me throughout my whole life.

ACKNOWLEDGMENTS

The present work was developed with the support of the Coordenação de Aperfeiicionamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

The author thanks CNPq for partially sponsoring this research.

The author also thanks the contributions of Fabio Porto, Kele Belloze, Eduardo Bezerra and, most importantly, of her advisors, Eduardo Ogasawara and Pedro H. González. All the given support, advice and observations were most appreciated and very important to the author's professional growth.

RESUMO

Avaliação Comparativa de Previsões de Séries Temporais Não-estacionárias

Pré-processamento de dados é um passo crucial para mineração e aprendizado a partir de dados, e uma de suas atividades principais é a transformação de dados. Esta atividade é particularmente importante no contexto de previsão de séries temporais já que a maioria dos modelos de séries temporais assume a propriedade de estacionariedade, i.e., propriedades estatísticas não mudam ao longo do tempo, o que na prática é a exceção e não a regra para a maioria dos conjuntos de dados. Existem vários métodos de transformação desenvolvidos para tratar a não-estacionariedade em séries temporais. Entretanto, a escolha de uma transformação que seja apropriada ao modelo de dados e à série temporal de uma aplicação em particular não é uma tarefa simples. Este trabalho fornece um estudo e uma análise experimental de métodos para transformação de séries temporais não-estacionárias. O foco deste trabalho é prover conhecimento relacionado ao tópico e uma discussão quanto às suas vantagens e limitações para com o problema de previsão de séries temporais. O conhecimento adquirido neste estudo foi encapsulado em um framework sistemático para análise, comparação e seleção de configurações transformação-modelo para previsão de séries temporais não-estacionárias. Um subconjunto dos métodos de transformação estudados é comparado através de uma avaliação experimental usando-se conjuntos de dados referenciais advindos de competições de previsão de séries temporais e outros conjuntos de dados macroeconômicos. Métodos de transformação de séries temporais não-estacionárias adequados forneceram melhorias de mais de 30% em acurácia de previsão para metade das séries temporais avaliadas e melhoraram a previsão em mais de 95% para 10% das séries temporais. Além disso, a adoção de uma fase de validação durante o treinamento de modelos permite a seleção de métodos de transformação adequados.

Palavras-chave: Não-estacionariedade; Séries temporais; Transformação de dados; Previsão; Framework

ABSTRACT

Benchmarking Nonstationary Time Series Prediction

Data preprocessing is a crucial step for mining and learning from data, and one of its primary activities is the transformation of data. This activity is very important in the context of time series prediction since most time series models assume the property of stationarity, i.e., statistical properties do not change over time, which in practice is the exception and not the rule in most real datasets. There are several transformation methods designed to treat nonstationarity in time series. However, the choice of a transformation that is appropriate to a particular data model and time series of an application is not a simple task. This work provides a review and experimental analysis of methods for transformation of nonstationary time series. The focus of this work is to provide a background on the subject and a discussion on their advantages and limitations to the problem of time series prediction. Knowledge acquired in this review has been encapsulated in a systematic framework for benchmarking and selecting adequate transformation-model setups for nonstationary time series prediction. A subset of the reviewed transformation methods is compared through an experimental evaluation using benchmark datasets from time series prediction competitions and other real macroeconomic datasets. Suitable nonstationary time series transformation methods provided improvements of more than 30% in prediction accuracy for half of the evaluated time series and improved the prediction in more than 95% for 10% of the time series. Furthermore, the adoption of a validation phase during model training enables the selection of suitable transformation methods.

Keywords: Nonstationarity; Time series; Transformation methods; Prediction; Framework

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF LISTINGS

LIST OF ABBREVIATIONS

AIC	Akaike Information Criterion
AICC	Corrected Akaike Information Criterion
AN	Adaptive Normalization
AR	Autoregressive
AR(1)	First Order Autoregressive Model
ARFIMA	Autoregressive Fractionally Integrated Moving Average
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BCT	Box-Cox Transform
BIC	Bayesian Information Criterion
CRAN	The Comprehensive R Archive Network
CWT	Continuous Wavelet Transform
DIF	Simple Differencing
DT	Detrending
DWT	Discrete Wavelet Transform
ELM	Extreme Learning Machines Network
EMD	Empirical Mode Decomposition
ETS	Exponential Smoothing State Space Model
FDIF	Fractional Differencing
FNLARMA	Fourier Nonlinear ARMA Models
FT	Fourier Transforms
GARCH	Generalized Autoregressive Conditional Heteroscedasticity Model
HHT	Hilbert-Huang Transform
HW	Holt-Winter's Exponential Smoothing
IMFS	Intrinsic Mode Functions
IPEA	Institute Of Applied Economic Research Of Brazil
KZA	Nonparametric Kolmogorov-Zurbenko Adaptive Algorithm
KZF	Kolmogorov-Zurbenko Filter
KZFT	Kolmogorov-Zurbenko Fourier Transform

LM	Linear Models
LMM	Linear Mixed Model
LT	Logarithmic Transform
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MAS	Moving Average Smoother
MLM	Machine Learning Methods
MLP	Multilayer Perceptron Network
MM	Min-max Normalization
MODWT	Maximum Overlap Discrete Wavelet Transform
MRA	Multiresolution Analysis
MSE	Mean Square Error
NMSE	Normalized MSE
NNET	Feed-forward Neural Network
PCT	Percentage Change Transform
PM	Time Series Pattern Mapping
PR	Polynomial Regression
RBF	Radial Basis Function Network
RFRST	Random Forest Regression
RMSE	Root Mean Square Errors
SARIMA	Seasonal ARIMA
SDIF	Seasonal Differencing
SM	Structural Models
SMAPE	Symmetric MAPE
SVM	Support Vector Machine
SW	Sliding Windows Technique
TF	Theta Forecasting
THW	Taylor's Double Seasonal Holt-Winter's Model
VAR	Vector Autoregressive
VMD	Variational Mode Decomposition
WFT	Windowed Fourier Transform
WPT	Discrete Wavelet Packet Transform
WT	Wavelet Transform

CONTENTS

Introduction

Adequate data preprocessing is an important activity in any application aiming at data analytics. It generally demands a long time and dedication (PYLE, 1999; HAN; KAMBER; PEI, 2011). The main objective of data preprocessing is ensuring the quality of data serving as input to applied learning methods and therefore avoid obtaining inaccurate and/or incorrect results and conclusions (OGASAWARA et al., 2010). Among the activities commonly performed during preprocessing, one can list data cleaning, feature and sample selection, outlier removal, normalization, and data transformation.

The data transformation activity becomes particularly important in the context of prediction (HAN; KAMBER; PEI, 2011; ESLING; AGON, 2012). Prediction is knowingly a crucial aspect to decision-making activities. The future states of information about a problem can massively impact on the success or failure of its solution. The time series analysis and its prediction are object of interest of many researchers due to increasing importance and applications in science, business and government (SALLES et al., 2015). The prediction context encompasses both problems of classification (prediction of discrete data) and regression (prediction of continuous data) (HAN; KAMBER; PEI, 2011; ESLING; AGON, 2012; BUZA, 2018). However, henceforth this work only focus on the problem of predicting numeric time series data through regression. For simplicity, this work may refer to prediction and regression interchangeably.

Although a great variety of time series prediction methods exists in literature (CHENG et al., 2015), many of these methods and the majority of works that handle time series assume that the available time series is stationary (GUJARATI, 2002). In a stationary time series, statistical properties, such as mean, variance and covariance, remain constant over time and in any sample of data (GUJARATI, 2002; SHUMWAY; STOFFER, 2017). However, in practice, it is observed that such properties are not constant in the majority of real-world time series, especially in socioeconomics (TSAY, 2010), where many of them are nonstationary. Thus, when observed the presence of nonstationarity in a time series, it is a usual approach to search for ways to transform them to achieve stationarity so that the known time series prediction methods can be applied.

There exist several transformation methods in literature for coping with nonsta-

tionarity in times series. However, the choice for an adequate method to a particular time series application is not a simple task. The analysis of their features and expected advantages is crucial. Some of the features that should be considered are their initial data assumptions (including different kinds of nonstationarity, linearity, and seasonality) and their intrinsic properties (mathematical transformation or computational algorithm). In this context, a thorough overview of different transformation methods for handling nonstationary time series and their respective features becomes particularly important. However, not many authors focus on studying transformation methods for nonstationarity treatment (YANG; ZURBENKO, 2010; CHENG et al., 2015).

Furthermore, there is a wide variety of models for time series prediction, each one having different properties and complexities, and many of them are generated by state-of-the-art machine learning methods (MLM). Still, none of them is a silver bullet for prediction of time series data. Additionally, the presence of nonstationarity leads to the possibility of exploring different data transformation and model fitting methods for obtaining predictions. The number of modeling alternatives and combinations may become very high. Finding an adequate transformation-model combination that solves a time series prediction problem is similar to solving an optimization problem (WOLPERT; MACREADY, 1997).

Performance evaluation of a transformation-model combination for time series prediction generally involves performing three different and consecutive tasks: (i) preprocessing, *i.e.*, applying transformation methods to a time series data; (ii) training, *i.e.*, finding adjusted parameters that fit a model to a (transformed) time series given as input; (iii) testing, *i.e.*, predicting subsequent values for the observed time series and comparing them against the actual ones by using an adequate error measure (OGASAWARA et al., 2009). With this purpose, one usually needs to partition the available time series into two sets, respectively, the training[1] and testing sets. This approach can be used to consistently evaluate the performance of a transformation-MLM combination and appraise its results and errors. Moreover, the prediction performance metrics of different transformation-MLM combinations can be comparatively analyzed in a benchmarking process. Such benchmarking process provides a way of assessing the relative quality of predictions and selecting adequate transformation-model combinations for a particular time series application.

[1] During training, it is a common practice to add a validation phase to measure the quality of fitted model. In such cases, the entire training set is partitioned into actual training and validation sets.

There are several works that present benchmarking frameworks and tools for MLM performance assessment such as the provided by Diebold and Mariano (2002), Eugster and Leisch (2008), Ramey (2013) and Kumar et al. (2016). There are also works developed to facilitate automatic time series prediction such as the provided by Hyndman and Khandakar (2008) and Moreno, Rivas, and Godoy (2018). Nonetheless, there are no works that propose and implement a systematic benchmarking framework that focus on (i) time series prediction; (ii) addressing nonstationary properties; and (iii) comparing and selecting adequate transformation-MLM combinations. This gap aggravates the already intricate problem of selecting adequate transformation-model setups for a particular nonstationary time series prediction application. Moreover, there are no works that focus on the study of different ways to coerce a time series into stationarity and their effects on univariate time series prediction.

This work targets the mentioned gaps and contributes by providing:

- A thorough review of nonstationary time series transformation methods for time series prediction organized in categories.
- A timeline of related works presenting the evolution of data transformation methods for nonstationary time series prediction grouped by their domain of application.
- A systematic framework for benchmarking transformation methods and models for univariate nonstationary time series prediction.
- A benchmarking and experimental analysis of representative transformation methods for the time series prediction problem.
- Use case examples of the framework usability for benchmarking transformation methods and MLM modeling.

The proposed benchmarking framework encapsulates the knowledge acquired through the review of nonstationary time series transformation methods. Moreover, the framework enables the application of this knowledge together with the predictive capabilities of the most commonly used MLM and linear models (LM). The application of user-defined transformations and/or models is also possible. The framework provides means of benchmarking nonstationary time series predictions. The results of benchmarking can be useful either for indicating demands for prediction improvement or for selecting adequate transformation-model combinations. The implementation of the framework is

within the version 5.0 of the R-Package *TSPred* (SALLES; OGASAWARA, 2018), which was made available worldwide.

The developed framework was used for performing the benchmarking and experimental analysis of the reviewed transformation methods. The goal is to provide a practical point of view regarding their advantages and limitations to the univariate time series prediction problem. According to the experimental evaluation conducted, suitable nonstationary time series transformation methods provided improvements of more than 30% in prediction accuracy for approximately half (130/262) of the evaluated time series. Accuracy improvements reached more than 95% for over 10% of the evaluated time series. This observed outcome suggests the need for considering these transformation methods and for comparing them during time series prediction. Additionally, the adoption of a validation phase for exploring different transformation methods generally led to selecting one of the top 5 most appropriate for a particular time series.

Besides this introduction, the remainder of this work is organized as follows. Chapter 1 provides concepts regarding nonstationarity in time series. It presents (i) a review of the most researched transformation methods for coping with nonstationary time series for the problem of prediction; (ii) a timeline of publications grouped by their domain of application of the reviewed transformation methods; (iii) a description of other relevant techniques for modeling times series; and (iv) a background of related tools for benchmarking time series prediction. Chapter 2 describes the proposed benchmarking framework and its implementation. Chapter 3 benchmarks different transformation methods and discusses their effects to the problem of prediction of nonstationary time series. Chapter 4 gives use case examples of the usability of the developed framework for benchmarking transformation methods with MLM modeling. Finally, Section 4.4 concludes.

1- Time series and nonstationarity

A time series is a sequence of observations of an object of interest collected over time. When observations are related to a single variable, a time series is referenced as a univariate one. Commonly, the behavior of a univariate time series is studied as a function of its past data (HANSSENS; PARSONS; SCHULTZ, 2003). Generally, one may consider a univariate time series X as a stochastic process, that is, a sequence of n random variables, $<x_1, x_2, x_3, \ldots, x_n>$, where x_1 represents the value assumed by the series at the first (oldest) time point and x_n represents the value of the series at the newest time point (ESLING; AGON, 2012; SHUMWAY; STOFFER, 2017). The length n of a time series X is represented as $|X|$ and a specific time series observation is referenced as, x_t, indexed in time by $t = 1, \ldots, n$.

Most methods applied for time series prediction assume that the behavior of a time series presents a level of regularity over time, which is generally approached with the study of the concept of stationarity (GUJARATI, 2002; SHUMWAY; STOFFER, 2017). The following sections formalize the different types of stationarity.

1.1- Strict stationarity

In a strictly stationary time series, the probabilistic behavior of every possible sequence of values $<x_{t_1}, x_{t_2}, \ldots, x_{t_k}>$ is equal to that of the time shifted sequence $<x_{t_{1+h}}, x_{t_{2+h}}, \ldots, x_{t_{k+h}}>$. Therefore, Equation 1 is valid for all $k = 1, 2, \ldots$, all arbitrary integer time points $t_1, t_2, \ldots, t_k$, all arbitrary numbers $c_1, c_2, \ldots, c_k$, and all possible time shifts $h = 0, \pm 1, \pm 2, \ldots$ (SHUMWAY; STOFFER, 2017).

$$P\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = P\{x_{t_{1+h}} \leq c_1, \ldots, x_{t_{k+h}} \leq c_k\} \tag{1}$$

Usually, the definition of strict stationarity is considered too strong for most applications. Such definition implies that all possible distribution functions for all subsets of a time

series must be in agreement with their counterparts in the shifted sequence for all values of h. This property is scarcely observed in most time series. Moreover, when handling a single dataset, the evaluation of strict stationarity is often not straightforward (SHUMWAY; STOFFER, 2017).

1.2- Weak stationarity

A more widely adopted version of stationarity, namely weak stationarity, gives a milder definition of the property. A weakly stationary time series, X, is a finite variance stochastic process such that: (i) the mean function, $E(x_t) = \mu_t = \mu$, is constant and does not depend on time t; and (ii) the autocovariance function, $\gamma(s,t)$, between x_t and the time-shifted time series value x_s depends only on the difference $|s-t|$ (GUJARATI, 2002; SHUMWAY; STOFFER, 2017).

In other words, a weakly stationary time series presents constant mean and variance, and its covariance function depends only on the time difference (YANG; ZURBENKO, 2010). These constraints are very important since they enable statistical inference to be drawn based on any sampled subset of a time series (HANSSENS; PARSONS; SCHULTZ, 2003). As in most works in literature, henceforth the term stationary refers to a weakly stationary process.

An example of a stationary time series may be visualized in Figure 1a, where one may observe mean and variance functions which are independent of time. It represents a first order autoregressive model (AR(1)), which is defined as in Equation 2, where α is a constant and $\omega_t \sim N(0, \sigma_\omega^2)$ (GUJARATI, 2002). In Figure 1a, $\sigma_\omega^2 = 2$, $\theta = 0.5$, and $\alpha = 0$. Since $0 < \theta < 1$, any relevant impacts of past observations eventually become negligible and do not affect the global behavior of the time series. It follows that this model presents constant statistical properties such as $E(x_t) = \mu_t = \alpha/(1-\theta) = 0$ and $VAR(x_t) = \sigma_\omega^2/(1-\theta^2) \sim 2$, hence being considered stationary.

$$x_t = \alpha + \theta x_{t-1} + \omega_t, \;\; 0 < \theta < 1 \tag{2}$$

1.3- Nonstationarity

If a time series X violates any of the constraints imposed by a stationary process, it is considered a nonstationary time series. Nonstationarity may manifest in many different ways. Generally, it implies that the mean and/or variance functions of a time series are non-constant and vary over time, that is, they are dependent on time t. The changes in mean and/or variance in time series are often due to deterministic trends, structural breaks, level shifts or changing variances (a condition known as heteroscedasticity). They can also be due to the presence of unit roots (HANSSENS; PARSONS; SCHULTZ, 2003). Figure 1(b-e) shows representative nonstationary time series.

A trended model might be considered the simplest form of nonstationary time series. This model represents a process that has stationary behavior around a deterministic trend. This trend shifts the mean of a time series causing it to increase or decrease over time. Commonly, the deviations of a systematic trend may be a stationary variable, known as a detrended variable, which may be analyzed instead of the original time series. In that case, usual stationary models are applicable (HANSSENS; PARSONS; SCHULTZ, 2003; YANG; ZURBENKO, 2010). A time series that presents this behavior is called trend stationary. One may write an example model of such time series by adding a deterministic linear trend to the AR(1) model presented in Equation 2. This model is defined as in Equation 3, where βt is the trend term and $\omega_t \sim N(0, \sigma_\omega^2)$ represents white noise. The mean function of the process $E(x_t) = \mu_t = \beta t$ varies over time, violating the stationarity constraints. The time series represented in Equation 3 may be observed in Figure 1b where one can see the linear increasing mean function. This model gives an example of a process which is nonstationary in mean.

$$x_t = \alpha + \theta x_{t-1} + \beta t + \omega_t, \;\; 0 < \theta < 1 \tag{3}$$

Nonstationarity in a time series may also be caused by structural breaks, that happen at specific points in time, usually due to environment changes. These structural breaks may eventually result in level shifts in a time series, which cause the mean function to be different for different portions of the series. In that case, a time series can be partitioned, and one can separately analyze each data portion with different statistical

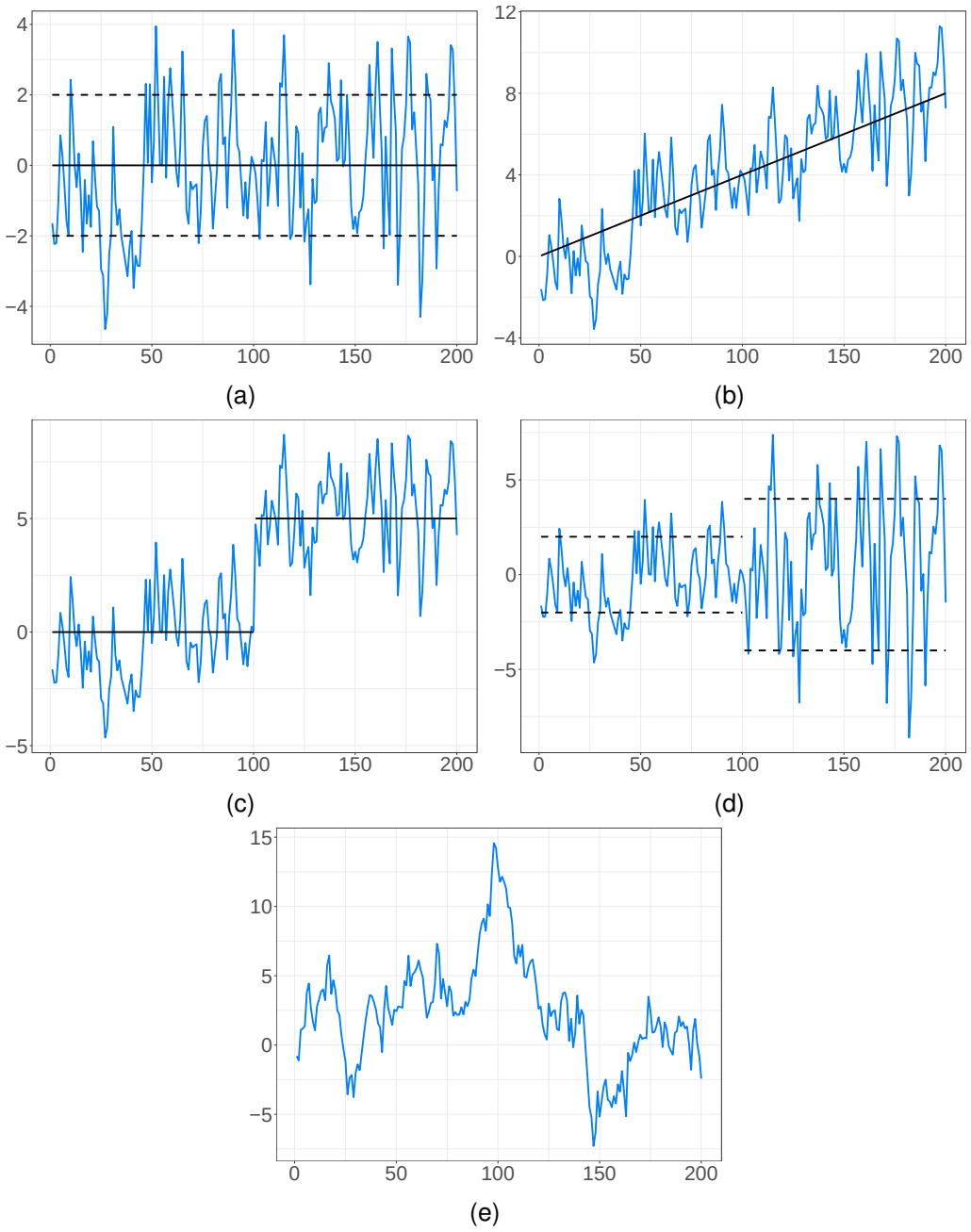


Figure 1 – Examples of time series presenting the properties of (a) stationarity, and non-stationarity in the form of (b) trend stationarity, (c) level stationarity, (d) heteroscedasticity and (e) difference stationarity. The solid and dashed black lines represent the mean and the variance functions of the time series, respectively.

properties, provided that the timing of a structural break is known (HANSSENS; PARSONS; SCHULTZ, 2003). Another way to handle structural breaks and level shifts in the mean function while modeling a time series is to make use of a dummy variable defined as zero before the point of a structural break and one after it. In case a time series presents local stationary properties on each different portion divided by a level shift it is known as being

level stationary. An example of a level stationary time series is observed in Figure 1c where one can see the level shifts of the mean function. The plot in Figure 1c represents the model in Equation 4 which is derived from Equation 2 by the addition of a dummy variable d_t, with the level shift $\delta = 5$ and the time of the structural break $t_b = 100$. This model gives another example of nonstationarity in the mean.

$$x_t = \alpha + \theta x_{t-1} + \delta d_t + \omega_t, \;\; 0 < \theta < 1, \;\; d_t = \begin{cases} 0, & t \leq t_b, \\ 1, & t > t_b. \end{cases} \tag{4}$$

Another cause of nonstationarity which results from structural breaks is the change in variance over time, a condition which is commonly known as heteroscedasticity (HANSSENS; PARSONS; SCHULTZ, 2003; SHUMWAY; STOFFER, 2017). Heteroscedasticity arises from environment changes that make the volatility of time series observations increase/decrease over time. Time series which present this condition are called heteroscedastic. Analogous to level shifts, the different variance properties in different portions of a time series can be addressed by partitioning the series or by modeling the changes in variance with a dependency on the structural breakpoints. An example of heteroscedastic time series is depicted in Figure 1d where different variance properties on the first and last portions of the series are easily observable. The series presented in Figure 1d represent the same model defined in Equation 2 and the same series of Figure 1a, but in this case, ω_t is set as $\omega_t \sim N(0, \sigma^2_\omega = 2)$ for $t = 1, \ldots, 100$ and $\omega_t \sim N(0, \sigma^2_\omega = 4)$ for $t = 101, \ldots, 200$. This model gives an example of a time series which presents nonstationarity in the variance.

An important type of nonstationarity, which in many cases is observed in real-world series, is caused by the presence of a unit root in the characteristic polynomial of a time series model. Without a unit root, time series observations tend to fluctuate around deterministic components such as a mean or a trend. Conversely, when a unit root is present, observations do not revert to a historical level and may wander in any direction. The presence of a unit root implies that the time series suffer from the influence of long-run components or stochastic trends (HANSSENS; PARSONS; SCHULTZ, 2003). In that case, the removal of a stochastic trend, usually done by the application of a process called differencing, is often helpful to coerce such time series to stationarity. For that reason nonstationary time series that present unit roots are also known as difference stationary (BOX; JENKINS; REINSEL, 2008). A difference stationary time series is presented in Figure 1e that represents the so-called random walk model, which can be formulated as

in Equation 5. This model assumes that the value of a time series x_t at a time t can be explained by the value of the series at the time $t-1$ plus a random movement represented by ω_t (SHUMWAY; STOFFER, 2017).

$$x_t = \alpha + x_{t-1} + \omega_t = \alpha + \sum_{i=1}^{t} \omega_i \tag{5}$$

The random walk model in Equation 5 is also derived from the AR(1) model in Equation 2 by defining $\theta = 1$. The definition of $\theta = 1$ implies that any impacts caused by past observations result in a permanent effect on the global behavior of a time series. In this case, the mean function of the process is not fixed, and the variance function tends to increase over time (HANSSENS; PARSONS; SCHULTZ, 2003) as can be observed by Equation 6. This model gives an example of a process which presents a unit root and is nonstationary both in mean and in variance (YANG; ZURBENKO, 2010).

$$VAR(x_t) = VAR(\alpha) + \sum_{i=1}^{t} VAR(\omega_i) = t\sigma_\omega^2 \to \infty \; as \; t \to \infty \tag{6}$$

It is also important to remark that frequently the dependence of a time series on past data may occur by multiples of some underlying seasonal lag S. In that case, a time series presents periodic components, and therefore, its statistical properties such as mean and variance may periodically change, creating a dependence on time t. This makes seasonality another particular form of nonstationarity, which is often found in time series (YANG; ZURBENKO, 2010).

Generally, any form of nonstationarity, if not adequately addressed, can have a relevant impact on time series prediction applications. Overlooking nonstationarity properties in a time series may lead to misleading statistical inferences and bad or unexpected prediction results. The next section presents a categorization for some of the most researched transformation methods prepared for application to nonstationary time series.

1.4- Nonstationary time series transformation methods

Nonstationarity poses challenges to time series prediction, especially since this property is pervasive in many real-world scenarios. As a consequence, several methods for statistical analysis of nonstationary time series have been developed (NACHANE; CLAVEL, 2008). This section presents a discussion of some of the most researched time series transformation methods for handling nonstationarity. Focus is given to the methods generally applied to the problem of time series prediction. For providing a better overview of the methods and for helping the discussion of their particular features it is introduced a general categorization presented in Figure 2.

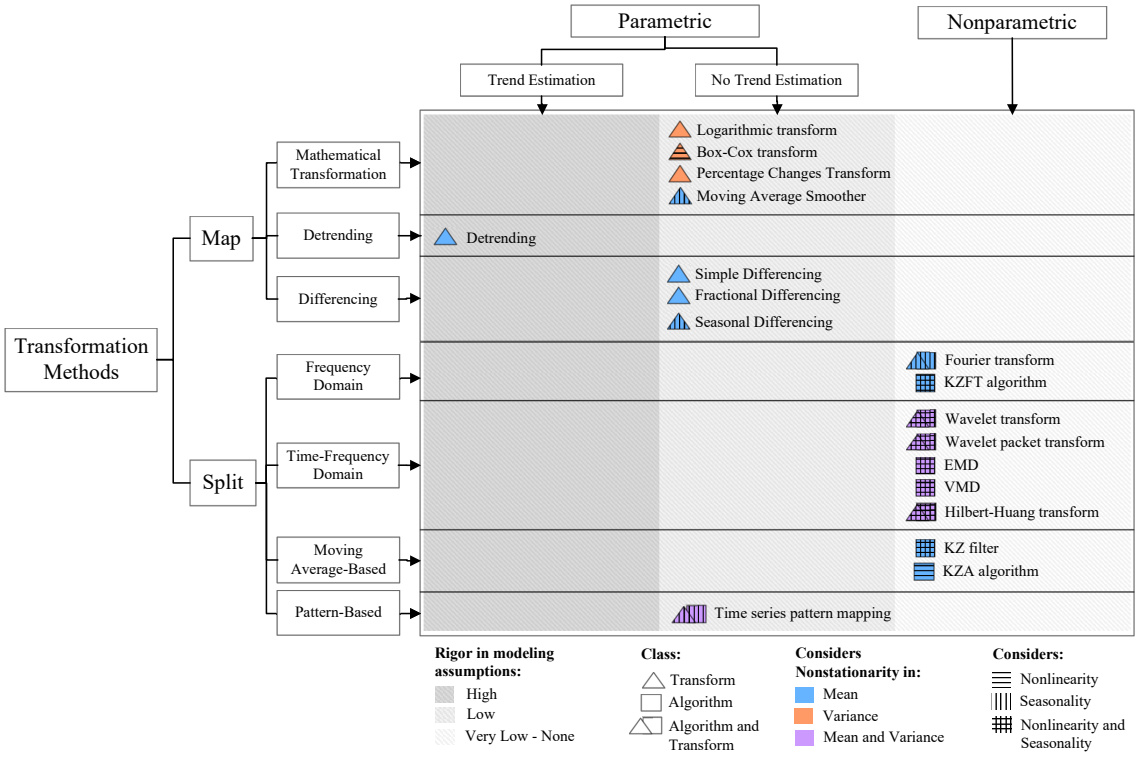


Figure 2 – Categorization diagram for some of the most researched transformation methods for coping with nonstationarity in univariate time series prediction.

1.4.1 Mapping and splitting classification of transformations

The diagram presented categorizes 18 of the most researched transformation methods for handling nonstationarity in time series by their two main classes: (i) mapping-based and (ii) splitting-based.

The mapping-based transformations correspond to methods that map the data of a time series to another representation. They return a new time series generated by passing each of its observations through a mathematical process. The transformed time series usually presents interesting properties, including stationarity, that are useful for the application of time series prediction methods that take them as input. The same parameters of the undergone transformation process may afterward be used for reversing it and obtaining time series data consistent with the original time series representation.

The splitting-based transformations refer to methods that are also able to produce a new representation of time series data. These methods apply different techniques for splitting a time series into a number of component series. Each component series generally serve as simpler or even stationary inputs for time series prediction methods that can analyze and predict them separately (DUDEK, 2016). The undergone decomposition process may afterward be reversed, generally by an additive or multiplicative combination of the component series, to obtain time series data consistent with the original representation. Splitting-based transformation methods are drawing increasingly more attention during the last few decades, especially when associated with computational intelligence techniques (CHIROMA et al., 2016). However, it might still be considered under-researched (NACHANE; CLAVEL, 2008; YANG; ZURBENKO, 2010).

1.4.2 Parametric and nonparametric methods

Within these two main classes, one can also categorize the methods into parametric and nonparametric according to their initial assumptions regarding the underlying behavior of a time series. Parametric methods make relatively more rigorous and inflexible assumptions about the parameters of the data distribution of a time series, whereas

nonparametric methods are data-driven and generally try to infer the number and nature of the parameters from the input time series (HAN; KAMBER; PEI, 2011).

From another view, parametric methods can be defined as having a fixed number of data parameters, while nonparametric methods allow for the number of parameters to grow with the amount of data in the training set. Although one can say that parametric methods make stronger assumptions about the nature of the data distributions of a time series, they are generally faster, straightforward to use, and easier to interpret. Moreover, although nonparametric methods present more flexibility, they can often demand an infeasible computational cost for large time series datasets and are prone to overfit the data (MURPHY, 2012).

Some parametric methods practice trend estimation. The estimated trend of a time series may be deterministic or stochastic. A deterministic trend is estimated as a deterministic function over time, which can be modeled by a standard regression model or a model of sine and cosine curves when a time series presents hidden cycles at different frequencies. Conversely, a stochastic trend is modeled as a nondeterministic function over time, such as an autoregressive moving average (ARMA) model, which usually serves as a stochastic component (YANG; ZURBENKO, 2010).

1.4.3 General settings and time series properties

The categorization presented in this work also classifies the transformation methods for handling nonstationary time series according to a set of features that may often be of interest to researchers, such as their class of implementation, the kinds of nonstationarity and the time series properties considered by the method. The implementation classes of the methods, being either a mathematical transform or a computational algorithm, are represented by the different geometric icons associated with each method. Some of the methods involve a time series transform that is performed by computational algorithms. In that case, their icons are formed by two of the respective geometric symbols.

The kinds of nonstationarity discussed in Section 1.3, namely nonstationarity in mean and in variance, which the transformation methods are prepared to handle, are also represented in the categorization diagram by the background color of the icon of each

method. Furthermore, some of the reviewed methods are also prepared to handle often crucial time series properties such as nonlinearity and seasonality. These methods and the properties they consider are represented in Figure 2 by the patterns (horizontal/vertical lines) which fill their respective icons.

1.4.4 Mapping-based transformation methods

Mapping-based transformation methods derive new mapped representations of a time series usually by undergoing one of three processes: (i) basic mathematical transformation, based on the use of standard mathematical operations and manipulation; (ii) detrending, based on the removal of a deterministic trend; or (iii) differencing, also based on the removal of a trend, but without the need for model estimation. The most commonly used mapping-based transformation methods are described in the next sections.

Basic mathematical transformations

Some of the most simple and the most applied nonstationary time series mapping-based transformations are the logarithmic transform (LT), Box-Cox transform (BCT) (SHUMWAY; STOFFER, 2017; CRYER; CHAN, 2010), percentage change transform (PCT), and moving average smoother (MAS) (MARROCU, 2006; OGASAWARA et al., 2010). The methods LT, BCT, and PCT are used for minimizing or even normalizing the variability over a time series. LT tends to suppress fluctuations that occur over portions of a time series which present higher values (SHUMWAY; STOFFER, 2017), and its simple formula can be seen in Equation 7, where $\hat{x}_t$ is the transformed version of the original time series value x_t. Often in macroeconomics, a time series is natural log-transformed (setting b $= e$) to minimize effects of nonstationarity and heteroscedasticity (non-constant variability (GUJARATI, 2002)), and also to induce symmetry and normality. The estimated coefficients of a logged time series can also be interpreted as elasticities. These

advantages make of natural logs one of the most applied LT (MARROCU, 2006).

$$\hat{x}_t = \log_{\mathsf{b}} x_t \tag{7}$$

The BCT, as in Equation 8, presents a more objective method for determining a suitable power transformation of a time series (YEO; JOHNSON, 2000). However, it demands the calculation of the numeric argument λ that is related to the data distribution function, which is not always known. Additionally, it can only be applied to positive valued data (CRYER; CHAN, 2010).

$$\hat{x}_t = \begin{cases} \left({x_t}^{\lambda} - 1\right)/\lambda, & \lambda \neq 0, \\ \log x_t, & \lambda = 0. \end{cases} \tag{8}$$

The PCT assumes that there is considerable stability in the relative percentage of change between two following observations of a time series (CRYER; CHAN, 2010). In that case, the Equation 9 is true if the percentage change is restricted to an interval $[-100\rho, 100\rho]$, with $0 \leq |\rho| < 1$ being a small acceptable threshold.

$$\hat{x}_t \approx \log\left(\frac{x_t}{x_{t-1}}\right) \tag{9}$$

Finally, MAS has been widely used especially in finance and econometrics. It is useful for highlighting seasonality and long-term trends in a time series (SHUMWAY; STOFFER, 2017; OGASAWARA et al., 2010). MAS can detect the evolving behavior of a time series by minimizing random noise (OGASAWARA et al., 2010), and can also be used for seasonal adjustment of a time series (MARROCU, 2006). One of the simplest forms of the MAS is represented by Equation 10, where k is the order of the moving average.

$$\hat{x}_t = \frac{1}{k} \sum_{i=t}^{t+k-1} x_i, \quad 1 \leq t \leq n-k+1 \tag{10}$$

Detrending versus differencing

It is also important to mention the transformations of detrending (DT) (or trend removal) and simple differencing (DIF), which are widely used in combination with time series modeling techniques. The DT transformation involves the determination and removal of an inherent trend observed in a time series behavior (SHUMWAY; STOFFER, 2017). Generally, the observed trend is estimated and defined by a deterministic functional form, which may be represented as a fixed component in a time series model (WU et al., 2007). A general DT transformation can be seen in Equation 11, where η_t is the determined trend component and, in this case, $\hat{x}_t$ represent the variability series, that is, the residue of the time series data after trend removal. However, this method usually assumes that the deterministic trend is always appropriate over time, which is frequently not the case in many applications, especially involving nonstationary processes (CRYER; CHAN, 2010). To overcome this drawback, recently there have been efforts towards data-driven, adaptive methods for determining trends in nonlinear and nonstationary time series (WU et al., 2007).

$$\hat{x}_t = x_t - \eta_t \tag{11}$$

The DIF method brings some advantages compared to DT since it does not demand a parameter estimation process and is capable of generating a stationary time series when it presents stationary behavior around a deterministic or stochastic trend (SHUMWAY; STOFFER, 2017). A first DIF transformation, simply written as in Equation 12, can eliminate a linear trend, a second DIF eliminates a quadratic trend, and so on. For DIFs of higher-order, the backshift operator (B) is used, as seen in Equation 13, where ∇^d is the d-th differencing and the operator $(1-B)^d$ is adapted for higher orders of d (SHUMWAY; STOFFER, 2017).

$$\hat{x}_t = \nabla x_t = x_t - x_{t-1} \tag{12}$$

$$\nabla^d = (1-B)^d,\ B^k x_t = x_{t-k} \tag{13}$$

It is also noted the variations of DIF, the fractional differencing (FDIF) and the seasonal differencing (SDIF). The FDIF method is a similar process to DIF. It allows for the possibility of d to assume the powers $-0.5 < d < 0.5$ (SHUMWAY; STOFFER, 2017)

and is commonly used for long memory time series analysis (JARA, 2011; MAYNARD; SMALLWOOD; WOHAR, 2013; SADAEI et al., 2016). In the SDIF method, d assumes the value corresponding to the seasonal period s, and it can be denoted as $\nabla^s x_t = x_t - x_{t-s}$ (CRYER; CHAN, 2010). However, a disadvantage of DIF is that the method does not produce an estimation of the inherent stationary process of a time series. In case this estimation is essential, DT may be more suited.

The DIF method can be used together with the linear ARMA models producing one of the most important time series linear prediction models, the autoregressive integrated moving average (ARIMA) models (BOX; JENKINS; REINSEL, 2008), which are able to model stochastic trends (CRYER; CHAN, 2010). An ARIMA(p, d, q) model is composed of an autoregressive (AR) and a moving average (MA) modeling processes (represented by p and q, respectively) with the application of a preliminary DIF process (I) (represented by the order d) so as to handle nonstationarity in time series (GUJARATI, 2002). ARIMA models assume that an observation of a time series, x_t, can be described as a function of its p past values and its q past white noise values (SHUMWAY; STOFFER, 2017). The latter are represented as ω_t, a Gaussian white noise series with mean zero and variance σ_ω^2. Let $\theta(B)$ and $\phi(B)$ be, respectively, the AR and the MA operators. The ARIMA model is denoted in Equation 14 (BOX; JENKINS; REINSEL, 2008; SHUMWAY; STOFFER, 2017). Variations of the ARIMA models may arise by introducing AR and MA polynomials that identify a dependence on some underlying seasonal lag, generating the seasonal ARIMA (SARIMA) models (SHUMWAY; STOFFER, 2017). It is also possible to allow d to take non-integer values, thus performing a preliminary FDIF process (FI), generating the autoregressive fractionally integrated moving average (ARFIMA) models (GIRISH; TIWARI, 2016).

$$\theta(B)(1-B)^d x_t = \phi(B)\omega_t. \tag{14}$$

1.4.5 Splitting-based transformation methods

Most commonly, splitting-based transformations are nonparametric and have its origin in signal processing techniques such as spectral (frequency domain) analysis (LOS, 2006; NACHANE; CLAVEL, 2008). They usually decompose a time series X into

components (signals) of the same length $|X|$ having different scales (frequencies), which may be caused by different physical phenomena, to capture the intrinsic dynamics of a time series (MORANA, 2007; YANG; ZURBENKO, 2010; SHU et al., 2014; XI et al., 2014; LAHMIRI, 2016). For example, a time series can be decomposed into short-term (high-frequency), seasonal, and long-term (low frequency) components. An advantage provided by this decomposition process is that the explanation of only a few kinds of signal components is generally simpler and more physically meaningful than a collection of estimated model parameters (SHUMWAY; STOFFER, 2017). Moreover, the derived components are more easily modeled, which can simplify the time series prediction problem (DUDEK, 2016). Usually, decomposition may be achieved in a frequency only or a time-frequency domain. The latter enables the preservation of information regarding localized changes. Special cases of time series decomposition are based on moving average iterations or pattern mapping (i.e., deriving patterns of a time series behavior to simplify its prediction and filter trends and long-term variations (DUDEK, 2016)). Some of the most frequently used splitting-based transformation methods are described in the following sections.

Frequency domain decomposition

Time series analysis in the frequency domain are often based on Fourier transforms (FT) (JOO; KIM, 2015). The FT creates a frequency-based representation (a spectrum) of a time series in terms of Fourier basis functions. The FT of a time series X, in this case represented as a function of time $x(t)$, can be formulated as in Equation 15, where $F(\xi)$ represents the Fourier spectrum, ξ is a frequency component and j is the imaginary number $j = \sqrt{-1}$ (LOS, 2006).

$$F(\xi) = \int_{-\infty}^{+\infty} x(t) e^{-j\xi t} dt, \;\; e^{-j\xi t} = \cos \xi t - j \sin \xi t \tag{15}$$

As well as for the Fourier series, the drawback of FT is that it is a global representation of a time series, and therefore information associated with time is lost (LOS, 2006; JOO; KIM, 2015). This fact makes the FT mainly suitable when a time series

does not present a relevant change in behavior over time, that is when it is stationary. Furthermore, FT does not perform well when handling large data, being limited to an interval between 0 and 2π observations (JOO; KIM, 2015). In order to minimize these limitations, a windowed variation of FT, namely the windowed Fourier transform (WFT), was devised. The WFT is suitable to analyze nonstationary time series with slowly changing behavior. Also, derivations of WFT are possible such as the Kolmogorov-Zurbenko Fourier transform (KZFT), which applies the WFT with a unique short-time window, being capable of providing the best possible resolution in the frequency domain (YANG; ZURBENKO, 2010). Nonetheless, WFT and its derivations may still generate time ambiguity, due to the infinite support of the Fourier wave bases (LOS, 2006). Recently, a method capable of efficiently analyzing localized changes in time series has been widely applied, which is based on finite wavelet bases (LOS, 2006).

Time-frequency domain decomposition

Wavelets are finite basis functions localized in both time and frequency. The wavelet transform (WT) decomposes a time series (signal) by correlating it with a family of wavelets, providing an extremely flexible time-frequency representation (LOS, 2006; LAHMIRI, 2016). The WT decomposes a time series X, which is again regarded as a function of time $x(t)$, into the wavelet series $\hat{x}(t)$ observed in Equation 16 (JOO; KIM, 2015). Here the components $\zeta(t)$ and its coefficient b represent the scale part of the wavelet series (being responsible for modeling trends and seasonality), whereas the components $\psi_(t)$ and its coefficient c represent the detail part of the wavelet series (corresponding to noise or random deviations), at scale (decomposition level) l and position k. Also, L is the defined maximum decomposition level.

$$\hat{x}(t) = \sum_{k=1}^{n} b_{l,k}\zeta_{l,k}(t) + \sum_{l=1}^{L}\sum_{k=1}^{n} c_{l,k}\psi_{l,k}(t) \tag{16}$$

Since finite wavelets are usually irregular and asymmetric (JOO; KIM, 2015), the WT is particularly suited for analysis of nonlinear, noisy and nonstationary time series, with rapidly changing behavior (LOS, 2006; AN et al., 2011; ROSHAN; GOPURA;

JAYASEKARA, 2011; LAHMIRI, 2016). If the wavelet transform is applied to a nonstationary time series, the resulting decomposed series usually present a better behavior than the original series, being more easily and accurately predicted, even by simple linear models like ARIMA. Predictions of the original time series with increased accuracy can then be obtained by applying the inverse WT to the predictions of the decomposed series (CONEJO et al., 2005; JOO; KIM, 2015). WT is also a tool for multiresolution analysis (MRA), that is, modeling frequency behavior of a time series at multiple time resolutions, which is an improvement over WFT (LOS, 2006). The MRA often makes of WT a more appealing option than standard econometric models (LAHMIRI, 2016). The WT, in MRA, can generally be implemented as a continuous wavelet transform (CWT) or a discrete wavelet transform (DWT) (CONEJO et al., 2005; AN et al., 2011). Prediction tasks, however, require a variation of DWT, called the maximum overlap discrete wavelet transform (MODWT) (NACHANE; CLAVEL, 2008). It is also noted the derived method of discrete wavelet packet transform (WPT), which reapplies the WT to a resulting decomposed series that does not present a required level of stationarity or decorrelation (ATTO; BERTHOUMIEU, 2012).

Besides the WT, there are other MRA techniques that are very recent and prominent, namely the empirical mode decomposition (EMD) and the derived variational mode decomposition (VMD) (LAHMIRI, 2016). Similarly to the WT, the EMD is an algorithm for nonlinear and nonstationary time series decomposition, generating a time-frequency representation of the series (LAHMIRI, 2016; WANG et al., 2016). EMD is also capable of decomposing a time series into more stable components (intrinsic mode functions (IMFs)), which can be more easily modeled, improving prediction accuracy (SUN et al., 2016). However, the advantage of EMD is that it does not depend on predetermined (wavelet) functions, since its basis functions are derived directly from the time series, therefore being adaptive and completely data-driven (LAHMIRI, 2016; WANG et al., 2016). Conversely, EMD is recursive and does not perform well when it comes to separating time series components with similar frequencies. In order to overcome this limitation, the VMD algorithm was devised, which transforms the time series decomposition into a non-recursive and variational model (LAHMIRI, 2016; SUN et al., 2016). The VMD is superior to EMD in the sense that it decomposes a time series more precisely and is more robust to noise (SUN et al., 2016).

Another recent noteworthy approach is presented by the Hilbert-Huang transform

(HHT), which combines the use of EMD with the application of the Hilbert transform for each of the resulting IMFs (SONG et al., 2017). The Hilbert transform is used for computing instantaneous frequencies enabling a frequency-time-energy (or amplitude) representation of a time series. The HHT is capable of accurately representing local time and frequency characteristics of nonlinear and nonstationary time series (SONG et al., 2017).

Moving average-based decomposition

Finally, mention is due to another example of a method for time series frequency decomposition, denominated Kolmogorov-Zurbenko filter (KZF) (YANG; ZURBENKO, 2010). The KZF is an algorithm based on iterations of a moving average filter with noise suppression. KZF results in long-term and short-term components, where the latter is generally fit well by an ARMA model. It is also resilient to non-equally collected and/or missing time series observations (YANG; ZURBENKO, 2010).

It is also possible for a time series to suffer from breaks in behavior over time, a characteristic that contributes to the presence of nonstationarity. Such breaks may be analyzed by a process of intervention analysis, which is a special application of time series decomposition. This process is performed by the nonparametric Kolmogorov-Zurbenko adaptive algorithm (KZA) by zooming in the KZF results in the areas near a break. The KZA is capable of detecting the time and size of the breaks in a time series even when it presents a high level of noise. It also supports break detection when the long-term component of a time series presents a nonparametric nature, which would cause traditional statistical methods to fail (YANG; ZURBENKO, 2010).

Pattern-based decomposition

A special case for time series decomposition is presented by the Time series pattern mapping (PM). PM is a preprocessing technique applied to simplify the prediction

of time series presenting multiple seasonalities (DUDEK, 2016). PM decomposes a time series into patterns of its behavior (not necessarily of the same length), which can eliminate the properties of nonstationarity in mean and in variance. It can also remove trend and seasonal cycles of long-term periods. The decomposed patterns of a time series may be used as input to computational intelligence techniques (DUDEK, 2016).

1.4.6 Research scenario of nonstationary time series transformations

The available literature on transformation methods for nonstationary time series was analyzed through a systematic mapping study using a query string involving the keywords “univariate”, “time series”, “nonstationarity”, “transformation”, “model”, and “prediction”. The query was executed on the Scopus database in September 2018 and returned 269 references.

Approximately 72 publications were selected to base the discussed review due to their relevance to the researched subject and impact (citation count). It has also been performed an additional snowballing search in Google Scholar mainly on the subject of wavelet transform for time series prediction due to its increasing importance to the current research scenario on nonstationarity. This resulted in the addition of 6 papers to base our presented review. The selected 78 publications are presented in the timeline of Table 1.

The publications are presented in Table 1 in a chronological way, to provide a better visualization of the evolution of the adopted methods for addressing nonstationarity in univariate time series. The methods adopted or approached by each publication are represented within parenthesis after each authors names. Each publication is classified by the main tasks performed by their chosen methods.

According to the categorization in Figure 2, are represented with different colors the publications which adopted mapping-based transformation methods, splitting-based transformation methods or that focus on other modeling techniques for treating nonstationarity. The publications are also grouped by their domain of application of the adopted methods, namely the statistical/natural sciences, socioeconomic/financial and industrial/business domains.

Table 1 – Timeline table of publications presenting some of the most researched methods for coping with nonstationarity in univariate time series. The publications are grouped by their domain of application and categorized by the main task of the applied methods (referenced within parenthesis).

Year	Statistical/Natural Sciences	Socioeconomic/Financial	Industrial/Business
1981	Hipel (1981) (BCT, ARIMA, SARIMA) Stensholt and Tjøstheim (1981) (ARIMA)		
1987			Sarma, Sinha, and Basu (1987) (DIF)
1992			Bhattacharya and Basu (1992) (DIF)
1996	Baillie (1996) (GARCH, FDIF, ARFIMA)		
2001	Maier and Dandy (2001) (DT, DIF)		
2002	Dittmann and Granger (2002) (FDIF)		
2003	D'Elia and Piccolo (2003) (BCT, ARFIMA)		Abraham and Balakrishna (2003) (GARCH, FDIF, ARIMA)
2004		Gil-Alana (2004) (FDIF, ARFIMA) Milionis (2004) (ARIMA)	
2005	Omtzigt and Paruolo (2005) (DIF) Gil-Alana (2005) (FDIF)		Conejo et al. (2005) (ARIMA, WT)
2006	K. a Ko and M. b Vannucci (2006) (ARFIMA, WT) K. Ko and M. Vannucci (2006) (ARFIMA, WT) Fryzlewicz, Sapatinas, and Rao (2006) (WT)	Fryzlewicz and Nason (2006) (GARCH, WT) Los (2006) (GARCH, FT, WT) Hendry (2006) (DIF) Marrocu (2006) (LT, BCT, MAS, DT, FDIF, SDIF) Gospodinov, Gavala, and Jiang (2006) (GARCH, FDIF, ARFIMA)	
2007	Haldrup and Nielsen (2007) (FDIF) Brockwell (2007) (ARFIMA) Morana (2007) (FT) Palma (2006) (MAS, DT, SM, GARCH, ARFIMA, WT)	Caporale and Gil-Alana (2007) (FDIF)	
2008		Nachane and Clavel (2008) (GARCH, FNLARMA, WT) Mills and Markellos (2008) (DT, SM, GARCH, FDIF, ARIMA, SARIMA)	
2009	Cavaliere and Taylor (2009) (GARCH) Otok and Suhartono (2009) (ARIMA, SARIMA)	Brandao and Nova (2009) (ARIMA, SARIMA)	
2010	Yang and Zurbenko (2010) (LT, BCT, DT, SM, DIF, FT, KZFT, KZA, KZF) Gourieroux and Jasiak (2010) (FDIF) Pai and Ravishanker (2010) (ARFIMA, FDIF)	Minu, Lineesh, and Jessy John (2010) (GARCH, WT) Ogasawara et al. (2010) (MAS)	
2011	Jara (2011) (FDIF)	Roshan, Gopura, and Jayasekara (2011) (WT)	An et al. (2011) (WT)
2012	Percival and Mondal (2012) (WT) Chilès and Delfiner (2012) (SM) Atto and Berthoumieu (2012) (WPT)	James and Murthy (2012) (DIF)	
2013	Maynard, Smallwood, and Wohar (2013) (FDIF)	Gao et al. (2013) (WT) Gil-Alana and Jiang (2013) (FDIF)	
2014	Ljung, Ledolter, and Abraham (2014) (ARIMA)	Claveria and Torra (2014) (ARIMA)	Stefanakos and Schinas (2014) (LT, BCT, DT) Shu et al. (2014) (ARIMA, FT)
2015		Joo and Kim (2015) (ARIMA, SARIMA, WT)	
2016		Wang et al. (2016) (HW, ARIMA, EMD) Sadaei et al. (2016) (ARFIMA) Lahmiri (2016) (WT, EMD, VMD)	Sun et al. (2016) (VMD) Girish and Tiwari (2016) (THW, ETS, TF, ARIMA, ARFIMA) Chiroma et al. (2016) (WT) Dudek (2016) (PM) Akpinar and Yumusak (2016) (HW, ARIMA)
2017	Nury, Hasan, and Alam (2017) (ARIMA, WT) Song et al. (2017) (HHT) Douc, Fokianos, and Moulines (2017) (GARCH)	Corona, González-Farías, and Orraca (2017) (DIF) Corona, Poncela, and Ruiz (2017) (DIF) Zhang, Lin, and Shang (2017) (EMD, ARIMA) Xiong, Shang, and Bian (2017) (EMD) Rodriguez (2017) (GARCH, FDIF) Ismail and Awajan (2017) (EMD, ETS, ARIMA, SM, TF, HW) Barba and Rodríguez (2017) (WT)	Liu, Gu, and Peng (2017) (ARIMA, BCT, HW)
2018	Xie, Bijral, and Ferres (2018) (ARIMA,DIF)	Zhang et al. (2018) (WT, SARIMA) Ngene, Mungai, and Lynch (2018) (FDIF) Caporale and Skare (2018) (ARFIMA, GARCH, FDIF)	Bokde, Feijóo, and Kulat (2018) (EMD, DIF) Li et al. (2018) (WT, ARIMA) Jamalmanesh et al. (2018) (ARIMA)

Mapping-based transformations only Splitting-based transformations only Both mapping and splitting-based transformations Other techniques only

Domains of application

The timeline in Table 1 present 30 publications with a statistical/natural sciences domain of application, which mainly comprehend the problems of time series prediction and/or statistical modeling. The works of Hipel (1981) and Chilès and Delfiner (2012), particularly, focus on problems regarding Geophysics and Geostatistics, respectively. Song et al. (2017) perform analysis of geographic spatiotemporal series. Maier and Dandy (2001), Otok and Suhartono (2009), Pai and Ravishanker (2010) and Nury, Hasan, and

Alam (2017) all work with environment variables. In particular, Otok and Suhartono (2009) aimed at modeling rainfall in Indonesia, and Nury, Hasan, and Alam (2017) focused on predicting future temperature measurements in northeastern Bangladesh.

One can also observe 32 publications with a socioeconomic (GIL-ALANA, 2004; MARROCU, 2006; CAPORALE; GIL-ALANA, 2007; OGASAWARA et al., 2010; CORONA; GONZÁLEZ-FARÍAS; ORRACA, 2017; CAPORALE; SKARE, 2018) and financial (FRY-ZLEWICZ; SAPATINAS; RAO, 2006; GOSPODINOV; GAVALA; JIANG, 2006; LOS, 2006; HENDRY, 2006; MILLS; MARKELLOS, 2008; NACHANE; CLAVEL, 2008; ROSHAN; GOPURA; JAYASEKARA, 2011; SADAEI et al., 2016; ZHANG; LIN; SHANG, 2017; RO-DRIGUEZ, 2017; ISMAIL; AWAJAN, 2017; NGENE; MUNGAI; LYNCH, 2018) domain of application. Joo and Kim (2015) conduct experiments using datasets with different origins including not only the financial market but also sales and airport demand, among others. The researches of Milionis (2004), Lahmiri (2016) and Corona, Poncela, and Ruiz (2017) handle bill, interest and inflation rates, respectively, while Gil-Alana and Jiang (2013) evaluates the purchasing power parity hypothesis in China-United States relations. The authors Minu, Lineesh, and Jessy John (2010) and Claveria and Torra (2014) approach other socioeconomic issues such as tourism demand and security. Furthermore, the papers of Brandao and Nova (2009), James and Murthy (2012), Gao et al. (2013), Wang et al. (2016), Xiong, Shang, and Bian (2017), Barba and Rodríguez (2017) and Zhang et al. (2018) present applications regarding traffic analysis.

Finally, other 16 publications are mentioned, with an industrial/business domain of application, which mainly refer to problems related to the energy market (SARMA; SINHA; BASU, 1987; BHATTACHARYA; BASU, 1992; CONEJO et al., 2005; AN et al., 2011; STEFANAKOS; SCHINAS, 2014; GIRISH; TIWARI, 2016; CHIROMA et al., 2016; AKPINAR; YUMUSAK, 2016; DUDEK, 2016; SUN et al., 2016; LIU; GU; PENG, 2017; JAMALMANESH et al., 2018). Among these publications, it is noted that Conejo et al. (2005), Girish and Tiwari (2016), Dudek (2016), and Jamalmanesh et al. (2018) focus on the electricity market, whereas Stefanakos and Schinas (2014) and Akpinar and Yumusak (2016) focus on the markets of marine fuel, and natural gas, respectively. An et al. (2011) and Bokde, Feijóo, and Kulat (2018) study wind farm power and speed, respectively. Special cases are presented by Abraham and Balakrishna (2003), Shu et al. (2014) and Li et al. (2018) which handle cargo transport, access loads and other time series problems relating industry and business.

Discussion on research evolution

From the timeline diagram in Table 1, one can observe that many of the applications presented by the selected publications fall into the statistical/natural sciences domain. This fact helps to suggest that the methods for handling nonstationarity in time series may still be widely found in statistical/theoretical researches. In that case, it is important to remark the demand for applying adequate methods for coping with nonstationarity in other real-world datasets, generally used in problems related to socioeconomics, financial market, and above all, industry. Fortunately, this seems to be the tendency since 2013, where most publications focus on real and non-theoretic applications.

Moreover, the analysis of Table 1 furthers the discussion of an increasing tendency regarding the use of splitting-based time series transformation methods. Although the timeline of the selected papers starts in 1981, the papers that apply methods that perform time series decomposition were published only since the year 2005. Furthermore, several of the papers that handle time series decomposition methods still apply them together with the more established mapping-based transformation methods. The amount of selected papers that focus solely on splitting-based transformations seems to increase since 2006. The highest concentration of such papers was published by the year 2016.

One may extend the observed late tendency to the use of nonparametric methods, which practically compose the category of splitting-based transformation methods. Since the growing tendency of works based on time series decomposition and nonparametric methods is relatively recent, it should suggest the need for further research on both subjects (NACHANE; CLAVEL, 2008; YANG; ZURBENKO, 2010; CHIROMA et al., 2016). Table 2 presents a synthesis of the selected papers based on the timeline diagram in Table 1.

Also, a highlight is given to the use of WT among the time series decomposition and nonparametric methods that have been most approached in recent years for business applications in time series prediction (LAHMIRI, 2016), having drawn unprecedented interest in its combination with machine learning methods to improve time series prediction accuracy (CHIROMA et al., 2016). In fact, 62.5% $(20/32)$ of the selected papers in Table 1 that apply time series decomposition methods approach the use of the WT in an experimental application. Also, it is noted that 25% $(8/32)$ of such papers are based on

Table 2 – Synthesis of the selected publications presenting some of the most researched methods for coping with nonstationarity in univariate time series.

Domain of application	Publication	Applied transformations		
		Mapping-based only	Splitting-based only	Both
Statistical/Natural Sciences	1981-2005	100%	0%	0%
	2006-2018	31%	25%	31%
Socioeconomic/Financial	1981-2005	100%	0%	0%
	2006-2018	53%	30%	17%
Industrial/Business	1981-2005	75%	0%	25%
	2006-2018	42%	33%	25%
All	1981-2005	93%	0%	7%
	2006-2018	47%	28%	20%

the use of another prominent decomposition method, namely the EMD, which has been increasingly adopted since 2016.

1.5- Time series prediction models

Besides the reviewed transformation methods, other efforts have been derived for addressing nonstationarity in time series data. Among them are statistical and state-of-the-art machine learning modeling approaches that are surveyed by Cheng et al. (2015) that presents a comprehensive review and a comparative analysis of models for time series of complex systems (presenting nonstationarity and nonlinearity). This section presents relevant models and techniques that are generally adopted for time series prediction. Some of the presented approaches feature means of describing nonstationary properties of time series.

1.5.1 Statistical models

When researching methods for nonstationary time series transformation, one may also come across other relevant and useful techniques for addressing nonstationarity in time series. These techniques rely on the use of nonlinear and nonstationary modeling of time series. In particular, there is a category of models which are designed for modeling a time series by describing it as statistical terms. Some of these terms directly model nonstationary behavior, such as an inherent trend or seasonality. These models may also be combined with the application of transformations such as detrending and/or differencing to achieve stationarity. Examples of models falling into this category are set by the structural models (SM), linear mixed model (LMM), Holt-Winter's exponential smoothing (HW), Taylor's double seasonal Holt-Winter's model (THW), exponential smoothing state space model (ETS), theta forecasting (TF), and the generalized autoregressive conditional heteroscedasticity model (GARCH).

Firstly, the SM is a representative model in this category. SM consist of terms, such as trend and seasonal. These terms may provide a straightforward interpretation of a time series behavior. A general SM is presented in Equation 17, where η_t is the trend term, χ_t is the cycle term, and ω_t is the error term. Trend terms may be stochastic and can be estimated using an ARIMA process.

$$x_t = \eta_t + \chi_t + \omega_t \tag{17}$$

SM models may also be considered special cases of state space models and can be represented as so (CLEMENTS; HENDRY, 2005). Their parameters can, therefore, be estimated by a Kalman filter (CLEMENTS; HENDRY, 2005; YANG; ZURBENKO, 2010). A basic example of a state space model is given in Equation 18, which consists of two parts: a so-called state equation (Equation 18a) and an observation equation (Equation 18b). The latter is added since $\dot{x}_t$ (in this case considered a state vector) is assumed not directly observable. Instead one can only observe x_t, which is a linear transformation of $\dot{x}_t$ with the addition of noise. Here A_t is a measurement matrix, ϵ is a coefficient matrix, and ω_t

and υ_t represent noise (SHUMWAY; STOFFER, 2017).

$$\dot{x}_t = \epsilon \dot{x}_{t-1} + \omega_t, \tag{18a}$$

$$x_t = A_t \dot{x}_t + \upsilon_t \tag{18b}$$

It is possible for time series data to present dependencies and serially correlated errors due to clustering, that is when observations can be divided into related subgroups (clusters). The LMM models belong to the category of mixed-effects models and are designed to address such dependencies among observations, which would typically make standard linear models inappropriate (FOX, 2015). It considers both fixed (deterministic) and random (stochastic) effects of time in a time series. In this case, fixed effects are often modeled using a deterministic trend model, and random effects are modeled by stochastic AR or ARMA structures (YANG; ZURBENKO, 2010).

There are several forecasting frameworks based on the classical model of exponential smoothing. A general exponential smoothing model is expressed as in Equation 19, where each time series value x_t, here obtained by the function of time $x(t)$, is based on the fitting function vector for the time series $f(t)$, the coefficient vector $\kappa(t)$ (T is the transpose operator) and a white noise $\omega(t)$. The function $f(t)$ depends on its past values and the transition matrix M (MOGHRAM; RAHMAN, 1989).

$$x(t) = \kappa(t)^T f(t) + \omega(t), \quad f(t) = M f(t-1) \tag{19}$$

Methods based on the exponential smoothing are applied in numerous applications due to their robustness and accuracy (GIRISH; TIWARI, 2016). Among them, it is noted the HW method, which can be used for nonlinear modeling of heteroscedastic time series (DUDEK, 2016). It estimates level, slope and seasonal terms at each time point (WANG et al., 2016; DUDEK, 2016). Despite taking a long processing time to determine a few parameters, it is able to represent trend, seasonality, and randomness in an effective way (AKPINAR; YUMUSAK, 2016). The THW adapts the HW to accommodate a second seasonality (GIRISH; TIWARI, 2016). The ETS model is based on an extended range of exponential smoothing, and it refers to its three basic terms: error, trend, and seasonality (GIRISH; TIWARI, 2016). The TF applies differencing and exponential smoothing altogether (GIRISH; TIWARI, 2016).

Another model commonly applied in finance for estimating heteroscedastic time series is the GARCH (NACHANE; CLAVEL, 2008). A time series may be explained as a GARCH(p,q) model by Equation 20a, where μ_t is a mean function. Particularly, the noise series ω_t is i.i.d. N(0, 1), so that the conditional distribution of the residual time series $\tilde{X}$, where $\tilde{x}_t = x_t - \mu_t$, is N($0$, σ_t^2) (CARMONA, 2013; MILLS; MARKELLOS, 2008). The conditional variance σ_t^2 is defined by Equation 20b with α and β as coefficients (CRYER; CHAN, 2010). Although GARCH models present a useful tool for treating nonconstant variability in time series, they are known for not being able to capture long memory properties and highly irregular behavior (STEFANAKOS; SCHINAS, 2014).

$$x_t = \mu_t + \sigma_t \omega_t \tag{20a}$$

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{p} \alpha_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \beta_j \tilde{x}_{t-j}^2 \tag{20b}$$

Finally, a noteworthy approach for addressing nonstationarity is through time-varying modeling. This approach is adopted by the FNLARMA. The FNLARMA basically estimates an ARMA model with deterministic time-dependent coefficients approximated by a Fourier series (NACHANE; CLAVEL, 2008).

1.5.2 Machine learning models

MLM have been used for nonlinear time series prediction in areas such as manufacturing systems, finance, health informatics, and energy grids, among other fields (CHENG et al., 2015). Since the models generated by MLM are universal approximators, that is, they can approximate any continuous function to an arbitrary precision, they can be beneficial to the problem of modeling nonstationary time series properties. Some of the most relevant MLM are the feed-forward neural network (NNET), multilayer perceptron network (MLP), extreme learning machines network (ELM), radial basis function network (RBF), support vector machine (SVM), and random forest regression (RFrst).

A neural network is a bioinspired computational approach for recognizing structural data patterns through neurons that are connected through synapses. The synapses have associated weights representing the relevance of the connection (HAYKIN, S. O., 2008; MOREIRA et al., 2018). Usually neural networks present a feed-forward architecture

(NNET) as presented in Figure 3 (HAYKIN, S., 1998). It uses the present time series value x_t, and its m predecessors $x_{t-m}, \ldots, x_{t-1}$ (in Figure 3, $m = 2$), to approximate the next value x_{t+1}. During a training process, approximation errors are backpropagated in order to adjust synaptic weights. A practical time series prediction process would involve the setup of neural network parameters, such as the number of input entries, hidden layers, neurons in hidden layers, etc (OGASAWARA et al., 2009). NNET with error backpropagation has been employed for nonlinear time series prediction, outperforming traditional statistical methods such as ARIMA in functional approximation. Perhaps a disadvantage of NNET is that it still assumes that the dynamics underlying a time series are time-invariant (CHENG et al., 2015). A fully connected NNET is referenced as MLP, and it is probably the most common network architecture currently in use. Particularly, it has been widely adopted in astronomy applications (MACHADO et al., 2016).

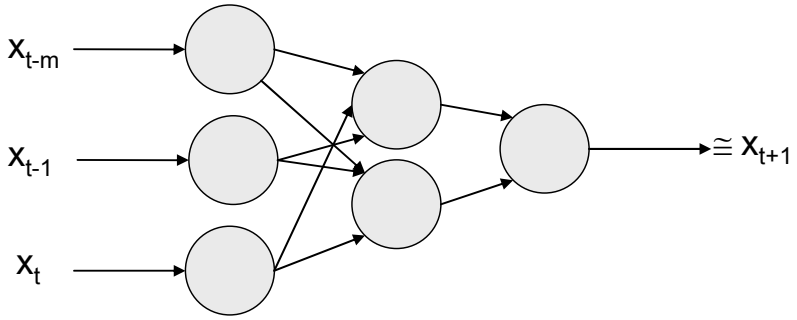


Figure 3 – Typical feed-forward neural network for time series

The slow learning speed of networks such as NNET and MLP has been a bottleneck in their applications. This fact is due to slow gradient-based learning algorithms, and iterative tuning of network parameters (HUANG; ZHU; SIEW, 2006). Unlike traditional implementations, the ELM network adopts a learning algorithm that randomly chooses hidden neuron nodes and analytically determines output weights. Thus, there is no need for any iterative tuning, or setting of parameters like learning rate, momentum, epochs, etc., making learning time very fast (HUANG; ZHU; SIEW, 2006).

An RBF network is similar to a MLP. However, it performs a linear combination of basis functions that are radially symmetric around a center/prototype (POGGIO; GIROSI, 1989). Several types of RBF networks, such as those that apply orthogonal least squares learning, and recursion were able to capture different forms of trends and volatility in time series (CHENG et al., 2015).

The SVM can be used for recognizing patterns in both linear and nonlinear data

(MOREIRA et al., 2018). Usually in a regression supported by SVM, a linear learning machine approximates a non-linear function in a kernel-induced feature space. The capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space (CRISTIANINI; SHAWE-TAYLOR, 2000; HAYKIN, S. O., 2008). SVM can have higher prediction accuracy than NNET and employ fewer parameters for chaotic time series prediction (CHENG et al., 2015).

RFrst is based on the combination of decision tree classifiers, or the ensemble of base models, that acts as a “forest”. After the formation of the forest, the model may combine the predictions of each tree additively or by average. The results are returned as the estimated time series prediction values (BREIMAN, 2001). The generalization error for a forest converges if there is a sufficiently large number of trees, which makes over-fitting not a problem (MOREIRA et al., 2018).

1.6- Time series normalization techniques

The process of normalization involves scaling time series data so that they fit into a new range. Although this transformation is usually not enough for addressing nonstationarity, data normalization is important for MLM algorithms since it increases their learning speed and prevents higher values to surpass the importance of smaller ones (RISSANEN, 2001; MOREIRA et al., 2018).

The Min-max normalization (MM) method is one of the most commonly adopted in literature (OGASAWARA et al., 2009). It applies a linear transformation to the data of time series X, where its minimum and maximum values (min_X and max_X) are used to transform each time series value x_t into $\bar{x}_t$ in the new interval [$\bar{min}_X, \bar{max}_X$], as shown in Equation 21 (MOREIRA et al., 2018). The main problem in using the MM method is that the minimum and maximum values of the out-of-sample dataset are unknown (OGASAWARA et al., 2009).

$$\bar{x}_t = \frac{x_t - min_X}{max_X - min_X} \cdot (\bar{max}_X - \bar{min}_X) + \bar{min}_X \tag{21}$$

Another approach commonly used for data normalization is the sliding windows technique (SW). Instead of considering the complete time series X for normalization, it

divides its data into sliding windows of length ω. It extracts statistical properties from X based on a fraction of ω consecutive time series values. Each window is normalized considering only these extracted statistical properties. This approach is based on the idea that decisions are usually based on recent data. SW always normalize time series data in the desired range, which is sometimes not possible using MM. However, it assumes that volatility is uniform, which is not usually true for nonstationary and heteroscedastic time series (OGASAWARA et al., 2010).

The Adaptive normalization (AN) approach was specially developed for nonstationary heteroscedastic time series (OGASAWARA et al., 2010). It is a variation of the SW, where the time series is transformed into a data sequence from which *global* statistical properties can be calculated. The *global* statistical properties are then used for normalization. The sliding windows data of AN are able to represent different volatilities (OGASAWARA et al., 2010).

1.7- Tools for benchmarking time series prediction

Another important issue researched in this work is the study and development of benchmarking frameworks and tools for MLM performance assessment in literature. In fact, we have observed that several authors focused on the task of benchmarking different models in many different domains. Ramey (2013) and Lessmann et al. (2008) developed frameworks for benchmarking classification models and algorithms. Moreover, Bischl et al. (2016) and Eugster and Leisch (2008) developed the R-packages *mlr* and *benchmark*, respectively, which provide tools for executing automated experiments of benchmarking a set of methods for data mining tasks such as classification and regression. These packages are mostly designed to support the use of data from relational models and focus on benchmarking based on plot visualization.

Hyndman and Khandakar (2008) and Hyndman et al. (2002) present frameworks for automatic forecasting using mainly linear methods such as ARIMA and ETS. Hyndman and Khandakar (2008) produced the well-known R-package named *forecast*, which can be used for automatic time series prediction. The R-package of Moreno, Rivas, and Godoy (2018), also facilitates time series prediction with the use of DIF, and BCT. Furthermore,

we note the works of Diebold and Mariano (2002), that proposes a variety of tests to compare the predictive accuracy of two different prediction methods, Diebold and Lopez (1996) that proposes an ensemble approach with the combination of different prediction methods, and Kumar et al. (2016) which propose a class of analytics systems to manage model selection using key ideas from data management research.

All in all, there are several works that present benchmarking frameworks and tools for MLM performance assessment. Nonetheless, to the best of our knowledge, there are no works that propose and implement a systematic framework for benchmarking MLM for time series prediction with focus on addressing nonstationary properties. This gap aggravates the complexity of the process of comparing and selecting adequate transformation-model setups for a particular nonstationary time series prediction application.

2- Benchmarking framework

This chapter presents the methodology for the implemented systematic framework for benchmarking transformation methods and MLM for univariate nonstationary time series prediction. The framework encapsulates the knowledge acquired with the review of nonstationary time series methods presented in Chapter 1. Implementation was done using R, which is both a language and environment with many statistical and graphical packages (R DEVELOPMENT CORE TEAM, 2008). The implementation of the framework composes part of the R-package *TSPred* (SALLES; OGASAWARA, 2018) for benchmarking time series prediction.

2.1- Time series prediction methodology

As an introduction to this methodology, a general univariate nonstationary time series prediction process is presented in Figure 4. It provides a systematic way of predicting a time series based on a particular setup of preprocessing methods and prediction model. The process depicted in Figure 4 also focus on prediction/model evaluation[1], that is, evaluation of the accuracy of prediction and/or the quality of fitness of a model. Such evaluation may indicate a demand for refining and perfecting the preprocessing-model setup and/or its parameters to obtain a more accurate model and lower prediction errors. This process may be repeated if the evaluated time series prediction setup does not reach a desired performance.

The first activity depicted in Figure 4 (labeled 1) refers to acquiring the training and evaluation (*i.e.*, either validation or testing) datasets, and performing data preprocessing by applying nonstationary time series transformation methods. Activity 2 refers to training the time series prediction model. The next activity (3) refers to applying the previously fitted model to an evaluation set to obtain predictions. Activity 4 corresponds to the

[1] Evaluation of prediction accuracy and/or model fitness can be performed either in a validation or a testing phase according to the target study.

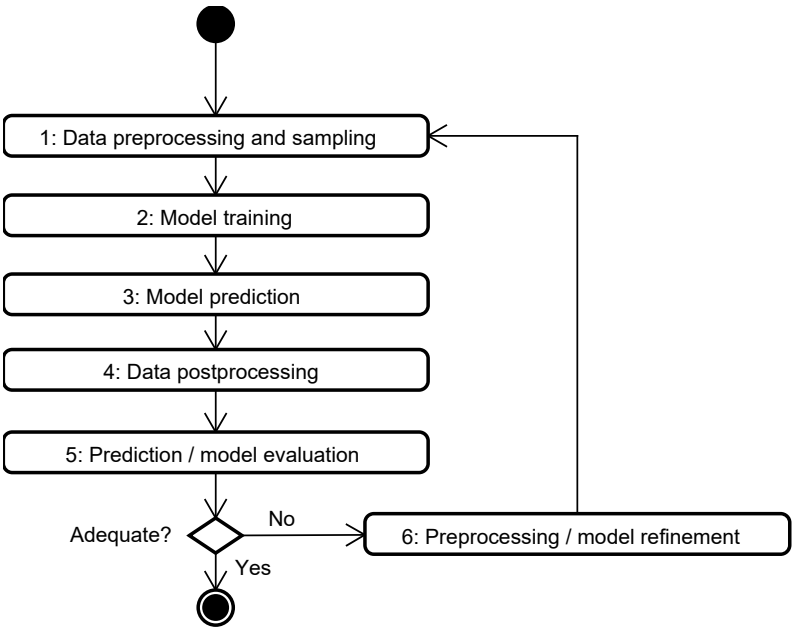


Figure 4 – Activity diagram for time series prediction methodology

postprocessing of predictions, reversing transformations applied to the time series data in activity 1. Finally, activity 5 is the evaluation of prediction errors yielded by the model, as well as model fitness metrics. If the results are not adequate, the transformation methods and the model can be refined, as indicated in activity 6. This entire process may be repeated if the prediction performance is not considered adequate. This process iteratively improves the quality of predictions.

In order to infer the adequacy of prediction results, it becomes important to adopt a benchmarking process. During a benchmarking process, given an input nonstationary time series, values predicted by a particular setup of preprocessing methods and prediction model (and their parameters) are compared to the ones found by other possible setups. Benchmarking provides a way of upraising relative quality of predictions and inferring adequate preprocessing-model setups for a particular nonstationary time series application. Furthermore, in order to adequately apply this assessment process and secure the validity of its results, it is important to adopt a reference (benchmark) preprocessing-model setup that includes a benchmark model that is well-established, reliable and widely adopted. One should also expect the benchmark setup to present a prediction performance good enough for creating a demand for refinements. Regarding this task, it is useful to adopt an easy to interpret benchmark model, since the analysis of the way in which the benchmark model is fitted may aid the configuration of other refined models. On the other hand, when no

appropriate benchmark setup is implemented, the other evaluated setups lack a reference to infer its practical value, and also the demand for adjustments may go undetected, and necessary improvements in accuracy and methodology may be neglected.

2.2- Framework structure diagram

This section describes the inner structure of the time series prediction benchmarking framework developed by this work. The main classes that represent the concept and structure of the framework are depicted in the class diagram in Figure 5. The main class of the framework is named *tspred*, standing for ”time series prediction”. An instance of this class represents a particular time series prediction application. This class encapsulates the process of Figure4 and all the elements (attributes) and activities (class methods) necessary to fulfill it.

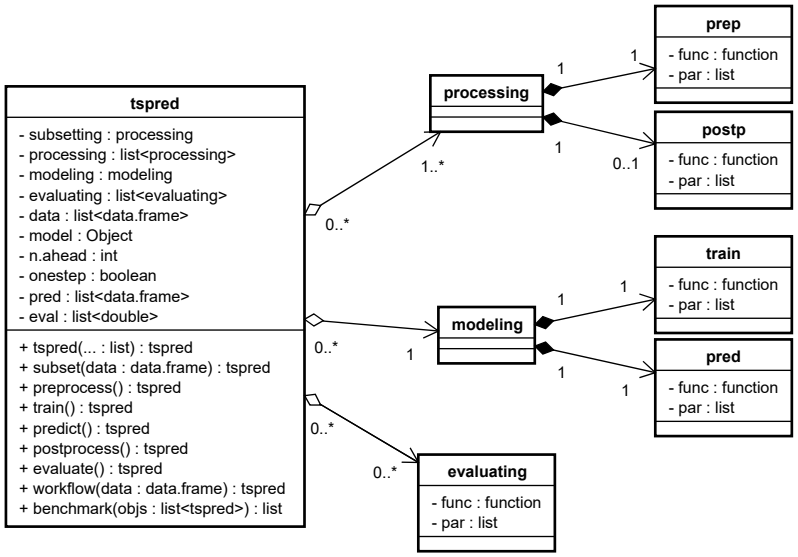


Figure 5 – Class diagram for the time series prediction benchmarking framework

The *tspred* class has three main parts, namely the *processing*, *modeling*, and *evaluating*. The *processing* class corresponds to general preprocessing and transformation methods (represented by the *prep* class) and their respective reverse transformations (represented by the *postp* class). Thus, the *processing* class encapsulates methods for performing the activities 1 and 4 of the diagram in Figure4. The *modeling* class corresponds to methods for training and predicting a particular time series model (represented

by the *train* and *pred* classes, respectively). Thus, the *modeling* class encapsulates methods for performing the activities 2 and 3 of the diagram in Figure4. Finally, the *evaluating* class corresponds to methods for evaluating quality of predictions and model fitness, encapsulating methods for performing the activity 5 of the diagram in Figure4. All *prep*, *postp*, *train*, *pred* and *evaluating* classes contain two main attributes, namely the function that implements their respective method (*func*) and a list of its parameters to be passed as input (*par*).

Generally the *tspred* class contains (i) one *processing* object for subsetting a time series into training and evaluation datasets (*subsetting* attribute); (ii) none or several *processing* objects for preprocessing/transforming and postprocessing/reverse transforming time series data (*processing* attribute); (iii) one *modeling* object for modeling and predicting a time series (*modeling* attribute); and (iv) none or several *evaluating* objects for evaluating the modeling and/or prediction of the given time series (*evaluating* attribute). Besides the aforementioned attributes, the *tspred* class also contains the input and output elements necessary for performing the activities in Figure 4. The attribute *data* contains all the time series data concerning a particular application. It contains the original time series which is given as input to (i); the resulting training and evaluation datasets which are given as input to (ii); and the resulting preprocessed/transformed datasets which are given as input to (iii). The attribute *model* contains the fitted model object generated by (iii). The attributes *n.ahead* and *onestep* are also passed to (iii) setting the prediction horizon and the type of prediction to be performed (multistep-ahead or one-step-ahead), respectively. The attribute *pred* contains predicted data, that is, the predicted data produced by (iii), which is given as input to (ii); and the resulting postprocessed/reverse transformed predictions, which are given as input to (iv). Finally, the attribute *eval* contains the evaluation metrics computed by (iv).

It is important to remark that the *tspred* class method is a constructor, therefore responsible for defining (instantiating) a particular time series prediction application. Nearly all class methods of *tspred* return a current *tspred* object with updated output elements and parameters. Particularly, the *subset* and *preprocess* class methods together perform the activity 1 of Figure4. While the *train*, *predict*, *postprocess* and *evaluate* class methods perform activities 2 to 5, respectively. Moreover, the class method *workflow* encompasses all activities from 1 to 5. Highlight is given to the class method *benchmark* that ranks the time series prediction applications described by a list of *tspred* objects based on the

collected evaluation metrics.

2.3- Framework implementation

Following the framework structure depicted in Figure 5, specific methods that help describe a time series prediction application (represented by the *tspred* class) can be implemented by extending the *processing*, *modeling*, and *evaluating* classes. This implementation design allows the user to define and apply any customized time series prediction methods according to demand. Nonetheless, the framework developed during this research includes the implementation of the main nonstationary time series prediction methods previously reviewed in this work. The implemented methods are observed in Figures 6, 7 and 8.

The main reviewed preprocessing and transformation methods for nonstationary time series were implemented by extending the *processing* class. The developed subclasses resulting from the extension (generalization) of the *processing* class can be observed in the diagram of Figure 6. All subclasses in Figure 6 form one generalization set that is complete and disjoint. The subclasses on the left of the diagram represent the nonstationary time series transformation methods LT, BCT, PCT, MAS, DT, DIF, EMD and WT, respectively. The subclasses on the right of the diagram represent other relevant preprocessing methods for nonstationary time series prediction with MLM. The *subsetting* subclass represents the activity of subsetting the time series data into training and evaluation datasets; the *SW* subclass represents the SW method for subsetting the time series data into sliding windows. The *NAS* subclass represents a method for handling NA values in time series data. The default behavior is to omit the NA values. Finally the *MinMax* and *AN* subclasses represent the MM and AN methods for normalizing time series data, respectively.

The methods for training and prediction based on the main reviewed models for nonstationary time series were implemented by extending the *modeling* class. The developed subclasses resulting from the extension (generalization) of the *modeling* class can be observed in the diagram of Figure 7. A second level of generalization was added in order to specify classes for statistical (*linear*) or machine learning (*MLM*) models.

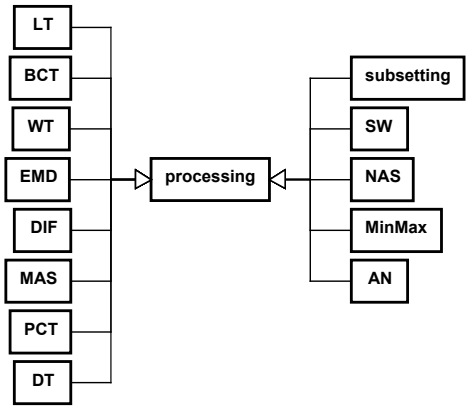


Figure 6 – Generalization of the *processing* class

The *MLM* class, in particular, contains attributes with *processing* objects for performing any necessary machine learning methodology tasks during the training and prediction activities. The *sw* attribute corresponds to a *SW* class object for coercing data into sliding windows, while the *proc* attribute corresponds to a list of *processing* objects for performing normalization and/or transformation of the data to serve as input for machine learning model training and prediction. Analogously, the subclasses in Figure 7 form one complete and disjoint generalization set. The subclasses of the *linear* class represent the models ARIMA, HW, TF, ETS and polynomial regression (PR), whereas, the subclasses of the *MLM* class represent the models generated by NNET, RFrst, RBF, SVM, MLP and ELM, respectively.

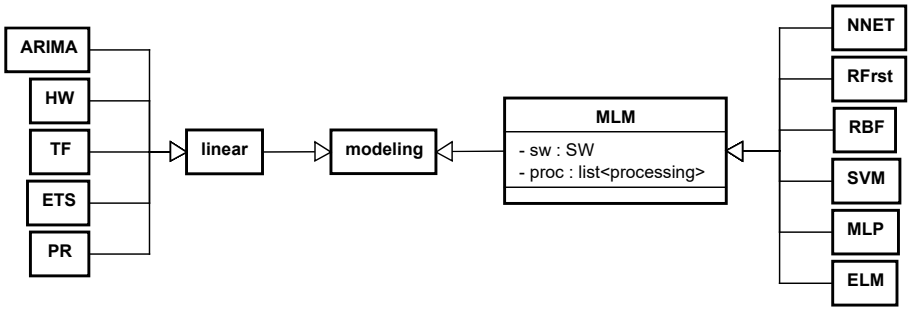


Figure 7 – Generalization of the *modeling* class

The methods for generating metrics for evaluating nonstationary time series prediction and modeling were implemented by extending the *evaluating* class. The developed subclasses resulting from the extension (generalization) of the *evaluating* class can be observed in the diagram of Figure 8. A second level of generalization was added in order to specify classes for prediction accuracy (error) measures (*error*) or model fitting criteria (*fitness*). Again, the subclasses in Figure 8 form one complete and disjoint generaliza-

tion set. The subclasses of the *error* class represent the prediction accuracy measures Mean Square Error (MSE), Normalized MSE (NMSE), root mean square errors (RMSE), Mean Absolute Percentage Error (MAPE), symmetric MAPE (sMAPE), and maximal error (*MAXError*), and the subclasses of the *fitness* class represent the model fitting criteria Akaike Information Criterion (AIC), corrected Akaike Information Criterion (AICc), Bayesian Information Criterion (BIC), and log-likelihood (*logLik*), respectively. A description of the implemented metrics can be found in the work of Davydenko and Fildes (2013).

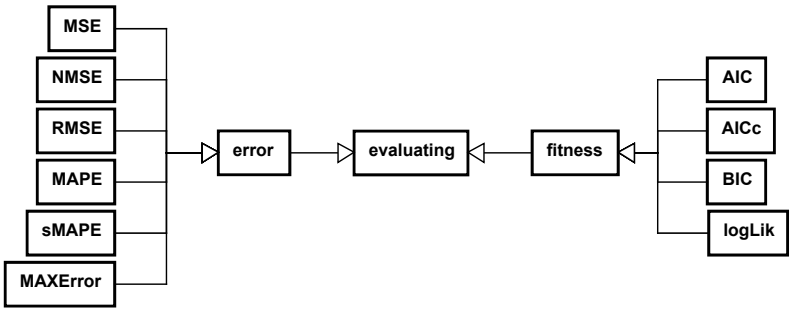


Figure 8 – Generalization of the *evaluating* class

2.4- The *TSPred* R-package

The implementation of the framework structure depicted in Figure 5 and all its subclasses was performed using the R programming language and environment and its provided package resources (R DEVELOPMENT CORE TEAM, 2008). The classes of the framework follow the S3 class system. The S3 system is very flexible and the most widely used in R programming language (WICKHAM, 2014). The framework is contained within the version 5.0 of the R-package *TSPred* (SALLES; OGASAWARA, 2018) for benchmarking nonstationary time series prediction. *TSPred* is freely available at The Comprehensive R Archive Network (CRAN) which hosts contributed packages for R from users worldwide. The package provides several automatized features that can be useful for any time series prediction application such as (i) transformation/model parameter selection; (ii) multistep-ahead or one-step-ahead prediction for both linear and machine learning models; and (iii) machine learning methodology tasks performed during training and prediction activities, among others. This work follows by giving some details

of the implementation of the nonstationary time series methods of this framework.

The transformation methods LT, BCT, PCT, and MAS were implemented as mathematical transforms following their characteristic equations. The BCT parameter (λ) is automatically obtained using the function *BoxCox.lambda* of the *forecast* R-package (HYNDMAN; KHANDAKAR, 2008). For the MAS method, a function was implemented for optimizing its order of moving average according to the best predictions from a cross-validation dataset. The transformation method DT was implemented and provides a fittest PR model for detrending the original time series before prediction. The DIF implementation can perform the tasks of either simple or seasonal differencing processes. It performs unit root tests (simple or seasonal) to determine the number of differences required for stationarity. This is done by the functions *ndiffs* or *nsdiffs* of the *forecast* R-package (HYNDMAN; KHANDAKAR, 2008). For the EMD implementation, functions were used from the R-package *Rlibeemd* (LUUKKO; HELSKE; RÄSÄNEN, 2016), while the WT implementation relied on the *wavelets* R-package (ALDRICH, 2013). Analogously to the MAS method, functions were implemented for optimizing the parameters of EMD (meaningful IMFs and boundary condition) and WT (filter and level of decomposition) according to the best predictions in a cross-validation dataset. Particularly for the WT, the MODWT transform was implemented.

Note is also given to the fact that all linear models of this framework, namely, the ARIMA, HW, TF, ETS, and PR were implemented by functions that automatically optimize their parameters. The ARIMA parameter optimization is done by conducting the Hyndman and Khandakar algorithm (HYNDMAN; ATHANASOPOULOS, 2013) using the *auto.arima* function of the *forecast* R-package (HYNDMAN; KHANDAKAR, 2008). Analogously, the HW, TF and ETS models are optimized using functions from *forecast* R-package. The PR parameter optimization is done by performing prediction tests with cross-validation. Regarding the machine learning models, the implementation of automatic training and prediction is done by functions of the R-packages *nnet* (VENABLES; RIPLEY, 2002), *randomForest* (LIAW; WIENER, 2002), *RSNNS* (BERGMEIR; BENÍTEZ, 2012), *e1071* (MEYER et al., 2018) and *elmNNRcpp* (MOUSELIMIS; GOSSO, 2018).

Evaluation metrics were mostly implemented according to their standard equations or by auxiliary functions from the *ModelMetrics* (HUNT, 2018), *MuMIn* (BARTOŃ, 2018) R-packages. For more details on the implementation of each nonstationary time series method of this framework, please refer to the documentation of *TSPred* (SALLES;

OGASAWARA, 2018).

2.5- Usage examples

This section gives examples for demonstrating the usage of the implemented framework for describing and performing a particular time series prediction application. The first example corresponds to a time series prediction using the ARIMA model, which can be considered a benchmark linear model for such applications (SALLES et al., 2017). In Listing 1 the *TSPred* R-package and the CATS dataset are loaded into the R programming environment. The object `tspred_arima` receives an instance of the *tspred* class. This instance counts with (i) a *subsetting* process for dividing a time series into training and evaluation datasets, where the latter has 20 observations; (ii) an ARIMA modeling process with automatic parameter selection; and (iii) a list of evaluation metrics to be computed for prediction accuracy (MSE) and model fitness (AIC). The prediction process represented by the object `tspred_arima` is applied for the third sequence of known values of the CATS series (`CATS[3]`) and executed by the function `workflow` that returns a *tspred* class object similar to `tspred_arima` updated with output elements and selected parameters `tspred_arima_res`.

The components of a time series process (implemented as subclass instances) in *TSPred* can be defined independently for enabling the reuse of subclass objects. An example of the definition of components/steps of a time series prediction process in *TSPred* is presented in Listing 2. First the *processing* objects are obtained for subsetting the time series and preprocessing it with the BCT and WT transformations. While parameters are set for WT, the parameters of BCT are automatically selected. Objects are also obtained for applying SW (with window length equal to 6 observations) and MM for normalizing the resulting windows of data during model training and prediction. Next a *modeling* object is obtained for representing a NNET modeling and prediction with 5 input units in a single hidden layer. `proc_sw` and `proc_mm` are given as parameters to the `modl_nnet` modeling object. At last an *evaluating* object respective to a MSE computation is obtained.

Listing 1 – R example for an ARIMA model prediction application using *TSPred*

```
#loading TSPred package
 > loadlibrary("TSPred")

#loading CATS dataset
 > data("CATS")

#defining the time series application
 > tspred_arima <- tspred(
     subsetting = subsetting(test_len = 20),
     modeling = ARIMA(),
     evaluating = list(MSE = MSE(),AIC = AIC())
   )

#performing the prediction application and obtaining results
 > tspred_arima_res <- workflow(tspred_arima, data = CATS[3])
```

Listing 2 – R example of the definition of components/steps of a time series prediction process in *TSPred*

```
#Obtaining objects of the processing class
 > proc_subset <- subsetting(test_len = 20)
 > proc_bct <- BCT()
 > proc_wt <- WT(level = 1, filter = "bl14")
 > proc_sw <- SW(window_len = 6)
 > proc_mm <- MinMax()

#Obtaining objects of the modeling class
 > modl_nnet <- NNET(size = 5,
                     sw = proc_sw,
                     proc = list(MM = proc_mm))

#Obtaining objects of the evaluating class
 > eval_mse <- MSE()
```

Listing 3 – R example for an MLM prediction application using *TSPred*

```
#Defining a time series prediction process
> tspred_mlm <- tspred(
    subsetting = proc_subset,
    processing = list(BCT = proc_bct,
                      WT = proc_wt),
    modeling = modl_nnet,
    evaluating = list(MSE = eval_mse)
  )

#Running the time series prediction process and obtaining results
> tspred_mlm_res <- tspred_mlm %>%
                    subset(data = CATS[3]) %>%
                    preprocess(prep_test = TRUE) %>%
                    train() %>%
                    predict(input_test_data = TRUE)  %>%
                    postprocess() %>%
                    evaluate()

#Benchmarking tspred objects
> bmrk_results <- benchmark(tspred_mlm_res,
                            list(tspred_arima_res))
```

The previously defined objects can then be referenced for defining a time series prediction process with an MLM represented by a *tspred* object (`tspred_mlm`) as can be seen in Listing 3. The execution of this process can be performed by the `workflow` function. Alternatively, each activity of the process can be executed separately or pipelined as showed in the example of Listing 3. The *tspred* objects containing results of ARIMA and NNET predictions (`tspred_arima_res` and `tspred_mlm_res`) can be benchmarked and ranked based on prediction evaluation criteria using the function `benchmark`.

A final example is given for the implementation of a user-defined MLM model. This can be done by extending the subclasses of the described structure of *TSPred* (Section 2.3), as displayed in Listing 4. Basically, the user needs only to define the functions for training and prediction of the model they wish to implement and their respective parameters. A *modeling* object representing the newly implemented model `my.model` can be obtained by using a function with the same name, giving parameter values as arguments.

Listing 4 – R example for the implementation of a user-defined MLM using *TSPred*

```
#Subclass my.model
> my.model <- function(train_par=NULL, pred_par=NULL){
    MLM(
      train_func = my.model.func,
      train_par=c(train_par),

      pred_func = my.model.pred.func,
      pred_par=c(pred_par),

      method="Name of my model",
      subclass="my.model"
    )
  }

#Obtaining an instance of the subclass my.model
> model <- my.model(train_par = list(par1 = a, par2 = b),
                    pred_par = list(par3 = c))
```

3- Benchmarking of transformation methods

The research on the various nonstationary time series transformation methods reviewed in this work pointed to the relevancy of an experimental comparison of their practical effects in the time series prediction problem. Such comparison may shed light on the advantages and limitations of these methods in practical applications. It can help researchers analyze their best options for treating nonstationarity. This work fills this demand by devising and conducting a benchmarking process and comparative analysis of eleven of the reviewed transformation methods that are most commonly used in practical applications. The developed framework described in Chapter 2 allowed the application of these methods in the prediction of time series of five different datasets originated from time series prediction competitions and real macroeconomic observations collected by a government institution. The datasets used in this experimental evaluation were made available [1] . The next sections describe the performed experiment in detail and discuss its results.

3.1- Datasets

Among the five time series datasets used in this experiment, three are benchmarks from time series prediction competitions (CATS (LENDASSE et al., 2007), NN3 (NN3, 2007), and NN5 (NN5, 2008)). The other two datasets are provided by the Institute of Applied Economic Research of Brazil (Ipea) (IPEA, 2017) and are derived from real economic and financial data of the world. When selecting these datasets, the aim was to obtain a reasonable number of representative time series presenting different types of nonstationarity and statistical properties to provide a discussion on the effects of the evaluated transformation methods applied to the prediction of a diverse range of time series.

Moreover, this choice of datasets was made so as to encompass all domains of

[1] https://github.com/RebeccaSalles/TSPred/wiki/nonstationary-review

application of the publications reviewed in Chapter 1.4.6. Particularly, the CATS dataset represents the statistical/natural sciences domain, the NN3 and NN5 datasets represent the industrial/business domain, and the datasets provided by Ipea represent the socioeconomic/financial domain.

The CATS Competition dataset presents an artificial time series with 5,000 observations, among which 100 are unknown. The unknown observations are grouped into five non-consecutive gaps of 20 successive values. The prediction of each gap may be considered a different problem, and each subset of the series followed by a gap may be considered a different time series to be modeled. In this context, the CATS dataset was considered as being composed of five time series of 980 observations. Both the NN3 and the NN5 Competition datasets present 111 time series. The series from the NN3 dataset have from 50 to 126 monthly observations drawn from a homogeneous population of real empirical business time series. All series from the NN5 dataset have 735 observations originated from daily withdrawals at 111 different cash machines within England, and may present missing data.

The two time series datasets provided by Ipea were selected as the most requested series collected in monthly and daily rates, and are henceforth referenced as Ipea_M and Ipea_D datasets, respectively. The Ipea is a public institution of Brazil that provides support to the federal government concerning public policies: fiscal, social, and economic. The data collected by Ipea and used in this experiment comprehend information on exchange rates (R$/US$), exports/imports prices, interest rates, minimum wage, unemployment rate, and more, measured from 1930 to September of 2017. Ipea_M contains 23 time series of 156 to 1019 observations. Ipea_D contains 12 time series of 901 to 8154 observations.

In order to obtain a better understanding of the statistical properties of the selected time series datasets, 7 of the most common statistical tests have been performed for autocorrelation, randomness and independence, heteroscedasticity, linearity, and stationarity. Table 3 contains a summary of the results of the statistical tests. For each dataset, it is presented the percentage of time series that had the null hypothesis test (H_0) confirmed.

The results in Table 3 show the considerable disparity in the statistical properties of the time series in the selected datasets. It is possible to observe that most time series did not confirm the null hypothesis of uncorrelated residuals and randomness. The heteroscedastic and linearity tests indicate that a substantial number of the time series present the properties of homoscedasticity and linear behavior around the mean. This

Table 3 – Statistical tests results and analysis

Statistical Tests	H0	CATS	NN3	NN5	Ipea_M	Ipea_D
Breusch-Godfrey	Uncorrelated residuals	0%	37%	0%	0%	0%
Box-Pierce	Randomness	0%	30%	0%	0%	0%
Goldfeld-Quandt	Homoscedasticity	40%	91%	48%	70%	50%
White Neural Network	Linearity in mean	100%	81%	59%	83%	75%
ADF[1]	Nonstationarity	100%	62%	5%	100%	58%
KPSS[2]	Trend Stationarity	0%	70%	71%	4%	8%
KPSS[2]	Level Stationarity	0%	57%	50%	4%	0%

[1]Augmented Dickey-Fuller
[2]Kwiatkowski-Phillips-Schmidt-Shin

result means that one can expect relative stability in the variance of the time series. Finally, the stationarity tests results show that a considerable amount of the time series over all datasets is nonstationary presenting unit root (also known as difference stationary), trend stationary (stationary around a deterministic trend) or level stationary (stationary around a level that changes over time). These latter results are particularly favorable for motivating the application of transformation methods such as the previously described in this work.

3.2- Experimental settings

Several predictions were performed in this experiment using the presented datasets. The prediction horizons used in these predictions were set according to each dataset. The CATS, NN3 and NN5 competitions recommend a prediction horizon for their datasets which were adopted in this experiments. Predictions were made for 20 observations ahead of the CATS times series (corresponding to the gaps of unknown values), 18 months ahead of the NN3 time series, and 56 days ahead of the NN5 time series. For the Ipea datasets, 12 months (a year) and 30 days (a month) were set as the prediction horizons for the Ipea_M and the Ipea_D datasets, respectively.

All predictions conducted in this experiment were performed using the ARMA models, defined as in Equation 14 (fixing $d = 0$). The use of ARMA models for prediction was combined with different preprocessing techniques performed by the nonstationary time series transformation methods listed in the next Section 3.3. The ARMA model

is a well-established linear model, and it is often applied as a benchmark method for time series prediction (SALLES et al., 2017). The choice of ARMA models is justified for two reasons: (i) for providing a fair comparison of our generated predictions and (ii) for allowing the focus of this experiment discussion to be on the comparison of different nonstationary time series transformation methods (and their effects on prediction) rather than on modeling performances.

All the experiments produced several model fitting criteria (AIC, AICc, BIC, and log-likelihood) and prediction accuracy measures (MSE, NMSE, MAPE, sMAPE, and maximal error (DAVYDENKO; FILDES, 2013)). Nevertheless, henceforth focus is placed on the MSE measurements to simplify the discussion of the prediction results, since it is generally the most commonly used measure for assessing prediction accuracy.

3.3- Implementation of nonstationary time series transformation methods

It was considered important to explore transformation methods of different categories of the presented review in this comparative analysis to provide a complete overview of the effects of the reviewed methods on prediction. Thus, among the reviewed nonstationary time series transformation methods, the eleven most commonly used in practical applications were selected for further analysis. For comparison purposes, it was also included in the experimental evaluation the naive approach of not applying any data transformation before prediction. Table 4 presents the selected transformation methods (and their reference acronyms) separated by the implemented data transformation process they apply and by their respective major categories. All selected transformation methods were implemented in the R-package *TSPred* (SALLES; OGASAWARA, 2018) (described in Chapter 2), including functions for automatically finding optimized parameters and for analyzing prediction results using different nonstationary time series methods. Particular implementation details of this comparative experimental analysis are given next.

All transformation methods referenced as DIF, SDIF, and DIFs perform the task of differencing a time series before modeling and predicting. The DIF, SDIF, and DIFs respectively correspond to the simple, seasonal, and combined simple/seasonal differencing processes. These methods are available within the ARIMA modeling process of

Table 4 – Nonstationary time series transformation methods selected for the experiment and the implemented data transformation process they apply

Category	Transformation	Method	Reference
Mapping-Based	Math. transform	Logarithmic transform	LT
	Math. transform	Logarithmic transform (base 10)	LT10
	Math. transform	Box-Cox transform	BCT
	Math. transform	Percentage Changes Transform	PCT
	Math. transform	Moving Average Smoother	MAS
	Detrending	Detrending	DT
	Differencing	Simple Differencing	DIF
	Differencing	Seasonal Differencing	SDIF
	Differencing	Simple and Seasonal Differencing	DIFs
Splitting-based	Decomposition	Wavelet transform	WT
	Decomposition	Empirical mode decomposition	EMD
None	None	None	Naive

the *auto.arima* function of the *forecast* R-package (HYNDMAN; KHANDAKAR, 2008). It performs KPSS tests to determine their differencing parameters.

Both EMD and WT perform the process of time series decomposition before the prediction. For the EMD, functions were used from the R-package *EMD* (KIM; OH, 2009). The decomposed time series were predicted separately by ARMA models in the case of WT. In the case of EMD, the IMFs were predicted by a vector autoregressive (VAR) model and the residue component (residue signal after extracting IMFs) was predicted by the fittest PR model. This latter definition is due to the implementation requests of the *EMD* package.

Particularly, for the method referenced as Naive, it implies that the naive approach was adopted and no data transformation was performed. In this case, the time series is directly predicted by an ARMA model. All code, experimental data, and results are made available [1] .

3.4- Results and discussion

This section discusses the results obtained with our comparative experiments for the selected nonstationary time series transformation methods. For that, it was used the MSE error measures produced by the predictions. The next subsections present

our analysis by focusing on the results generated over a single time series (subsection 3.4.1) and over all time series of each selected dataset (subsections 3.4.2 and 3.4.3). A discussion is also conducted around the results across all datasets (subsection 3.4.4) and the observed impact caused by transformation methods on prediction accuracy (subsection 3.4.5).

3.4.1 Analysis over a single time series

Taylor diagrams (TAYLOR, 2001) have been produced for summarizing in a single plot the different aspects of the effects of each transformation method in the prediction of a particular time series. Two time series of the CATS dataset were analyzed, as it is one of the most nonstationary datasets among the selected. The diagrams are depicted in Figure 9.

Figures 9a and 9b present prediction results for the time series corresponding to the second and third sequences of known values (1001-1980 and 2001-2980) of the CATS times series. Taylor diagrams illustrate the quality of predictions against reference values, which in this case are the real observations of the time series (that were to be predicted). Data regarding the real observations of the time series, as well as the predictions corresponding to the use of each nonstationary time series transformation method, are represented in the diagram by points inside a quarter circle, where the former is labeled as “observed”. The sections of the quarter circle correspond to the level of correlation between the predicted and observed values. The dashed brown contours centered in the observed point represent the level of RMSE of the predictions. The distances to the origin correspond to the standard deviation of the predictions, and the dashed black contour centered in the origin mark the standard deviation of the observed values. Small distances from the observed point mean better predictions. Particularly for our plots in Figure 9, the legends also present the methods ranked by smallest MSE errors.

The predictions in the Figure 9a present higher correlation and closer standard deviations to the observed values, besides smallest RMSE errors. Furthermore, the distances to the observed point seem to produce a similar ranking to the one in the

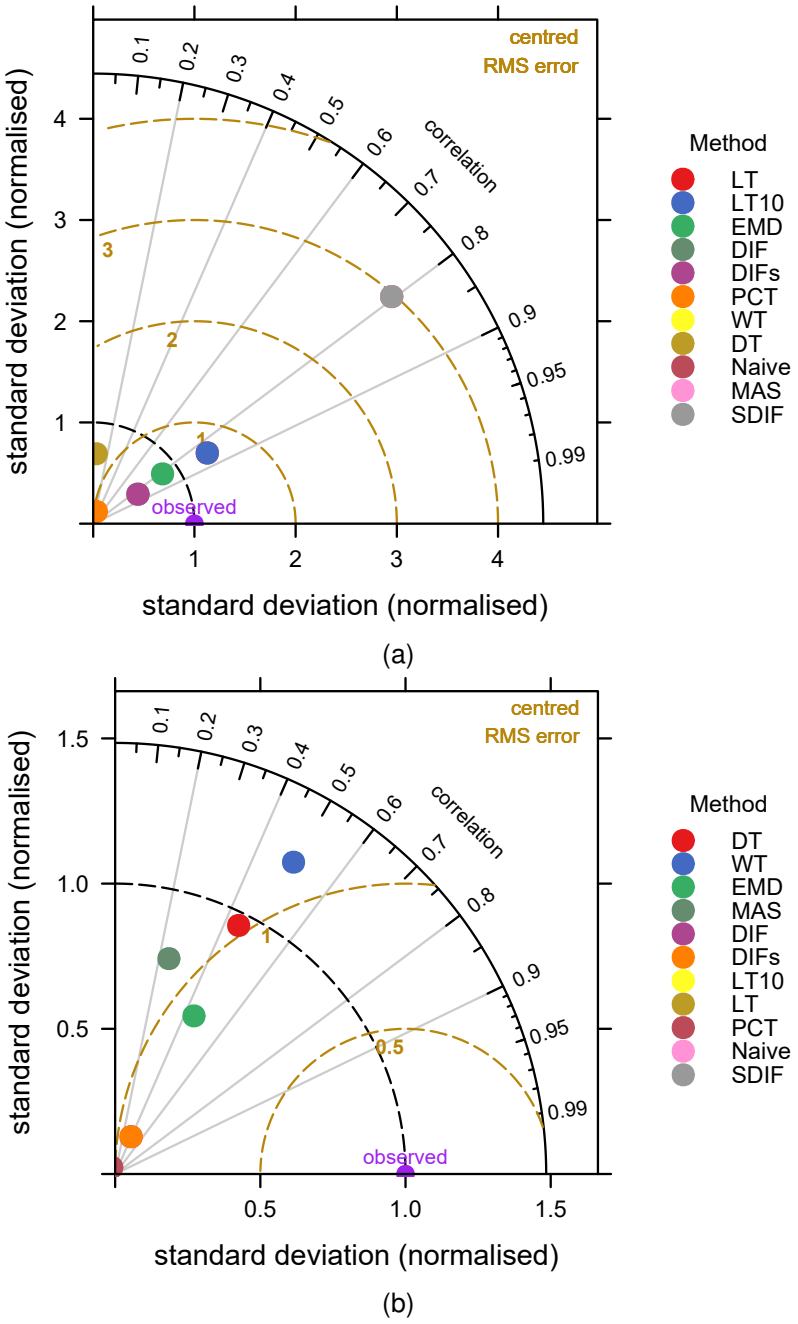


Figure 9 – Taylor diagrams of predictions for two CATS time series preprocessed by different nonstationary time series transformation methods. (a) presents results for the second sequence of known values 1001-1980 of CATS and (b) presents results for the third sequence of known values 2001-2980 of CATS

legend, based on the MSE errors. Conversely, the prediction results in Figure 9b make the identification of the best transformation method (for simplicity, best method) less clear. Although the results from EMD are closer to the observed, the results of DT present a more similar standard deviation, and the results of WT a higher level of correlation. Moreover, the distances to the observed point suggest a different ranking than the one in

the legend, even when only the RMSE error is considered. That indicates the need for special attention when selecting the error measure to be used during analysis since the assessment of the best method was not consistent for both measures.

3.4.2 Analysis over a single dataset

It was also considered important to summarize prediction results over all the time series in a particular dataset. For that, pairwise Wilcoxon statistical tests were performed for comparing the MSE errors yielded from predictions corresponding to all nonstationary time series transformation methods evaluated against each other. A visualization of the number of statistically significant prediction improvements provided by the use of each nonstationary times series transformation method over others in the experiments for the CATS dataset is presented in Figure 10a. Based on this plot, one may consider DIF, DIFs, EMD, and MAS among the best methods. The use of the Naive approach and the SDIF transformation method did not provide statistically significant prediction improvements over other methods. This was an expected result since all time series in the CATS dataset are nonstationary. The use of SDIF derived prediction performance (for short, performance) equal to the Naive method since no seasonal parameters were estimated by the Hyndman and Khandakar algorithm (HYNDMAN; ATHANASOPOULOS, 2013), suggesting a low level of seasonality in the series. Also, the DIF and DIFs methods both represent the method DIF for the same reason.

Figure 10 also presents another indicator that may be used for assessing the quality of the predictions. Namely, the number of times the use of each nonstationary times series transformation method yielded one of the five smallest absolute values of MSE prediction errors across all time series. That is, the number of times the use of each transformation method was responsible for one of the top 5 results of the time series in CATS. This information is presented in Figure 10b.

The time series of the CATS dataset presented considerable nonstationarity in the form of difference stationarity, which motivates the application of stochastic trend removal or differencing, which uses the operator in Equation 13, to achieve stationarity. This fact favored the transformation method DIF, which produced the best predictions, followed by

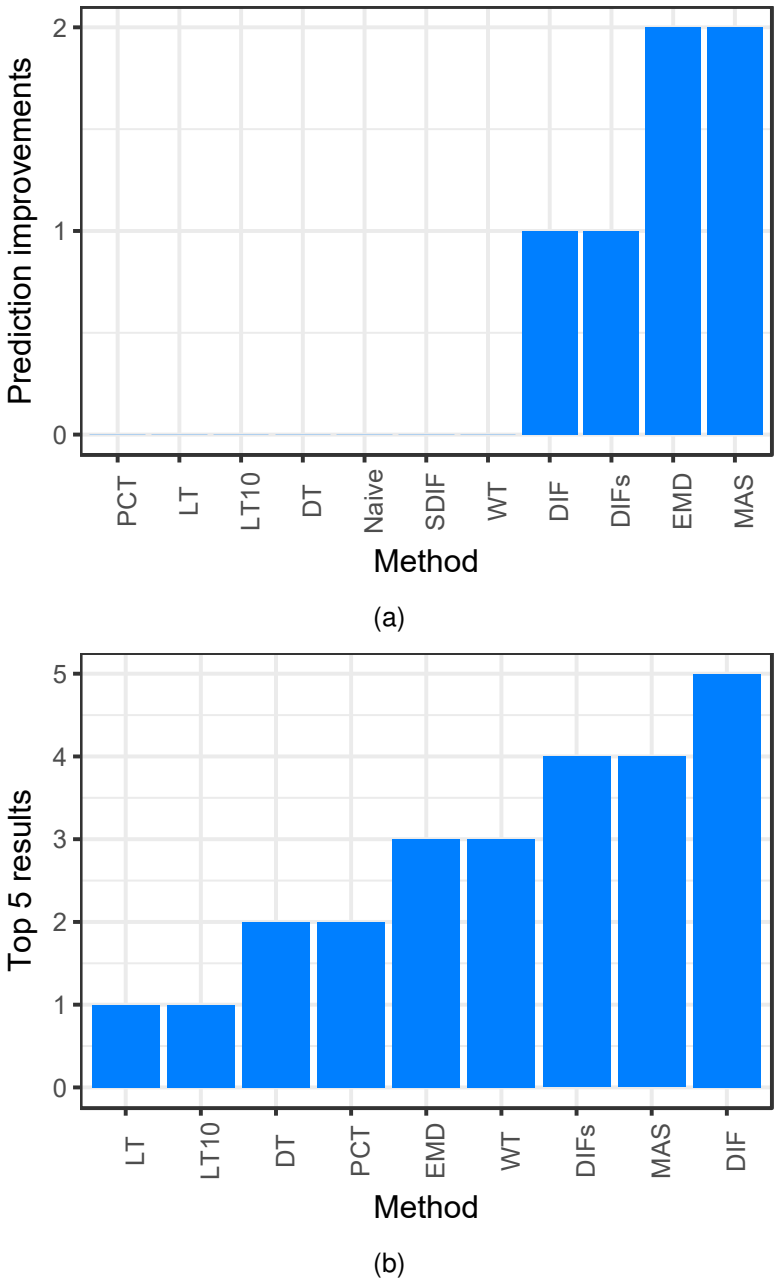


Figure 10 – Plots of the results for the prediction of all CATS time series preprocessed by each selected nonstationary time series transformation methods. (a) presents the number of statistically significant prediction improvements provided by the use of each nonstationary times series transformation method over others. (b) presents the number of times the use of each transformation method was responsible for one of the top 5 prediction results of the time series.

the DIFs, EMD and WT, as can be observed in Figure 10b. The results of MAS were again among the best, even though there was no significant seasonality as confirmed by the Hyndman and Khandakar algorithm. This may indicate the presence of clear long-term trends in the series generated by Equation 10.

Furthermore, the CATS time series presented no trend or level stationarity, which indicates that the removal of a deterministic trend may not be enough for providing stationarity. This fact helps explain the results obtained by the use of DT. Another important property of the CATS time series is that they presented the largest rate of heteroscedasticity (60%) among all datasets. The heteroscedasticity in the series possibly favored EMD and WT that are prepared to handle non-constant variability. It is also true for the PCT, LT, LT10, and BCT methods. The application of these latter transformation methods, however, was impaired due to the presence of negative data in all CATS time series, which caused the PCT, LT, and LT10 to produce NA values. Due to that, necessary parameters could not be computed for BCT in any time series, therefore not being present in the plots. This behavior is consistent with the Equations 7-9. Finally, highlight is given to the performances yielded by the use of both EMD and WT as the splitting-based time series decomposition methods evaluated in this experiment.

3.4.3 Analysis of results on each selected dataset

Next, the results of the experiments for each selected dataset are presented in Figure 11. The presented scatter plots summarize the same indicators presented in Figure 10 for all CATS, NN3, NN5, Ipea_M, and Ipea_D datasets. That is, they summarize (i) the number of times the use of each nonstationary time series transformation method provided a statistically significant prediction improvement over others and (ii) the number of times the use of each method was responsible for one of the top 5 prediction results of the time series (based on the MSE prediction errors). By analysis of the results depicted in Figure 11, it was defined that the use of a transformation method produced better predictions if (i) it was responsible for one of the top 5 results for the majority of time series in a dataset, or (ii) it provided at least half of the maximal number of statistically significant improvements in a dataset. Conversely, the transformation methods that led to worse predictions were highlighted in Figure 11 with a dashed rectangle. As the results regarding the CATS dataset were previously examined, henceforth focus is given to our discussion on the remaining datasets.

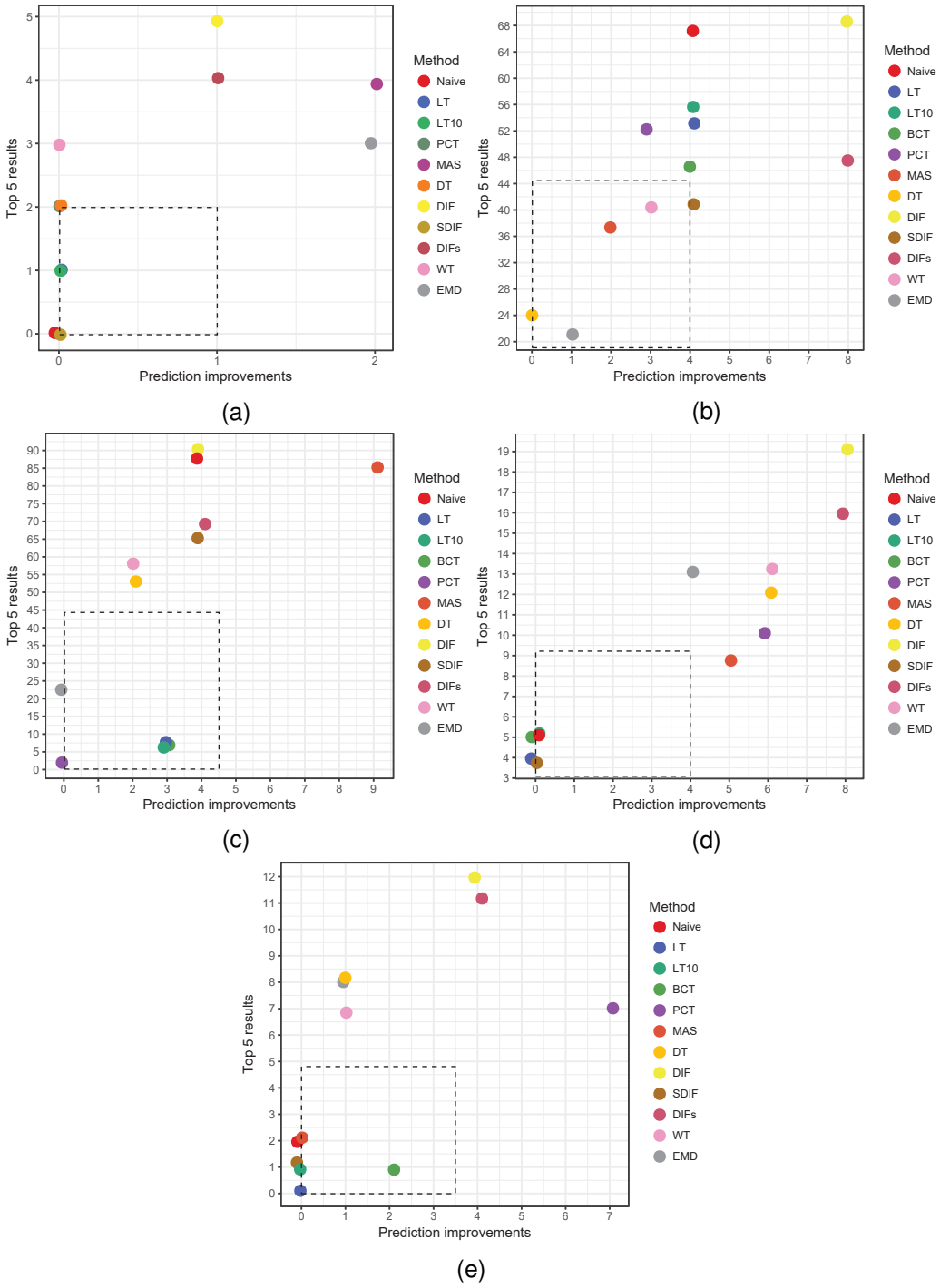


Figure 11 – Scatter plots for summarizing the results indicators shown in Figure 10 for all time series in the datasets (a) CATS, (b) NN3, (c) NN5, (d) Ipea_M and (e) Ipea_D. The dashed rectangles mark the transformation methods that led to worse predictions.

Results on NN3

The results for the NN3 dataset are presented in Figure 11b. Most time series of the NN3 dataset (62%) were difference stationary, also favoring the transformation

methods DIF and DIFs. It was similar to the results for the CATS dataset. Particularly, DIF and DIFs provided statistically significant prediction improvements over eight of the twelve evaluated transformation methods. However, surprisingly, the results produced by the use of the Naive approach were among the best despite nonstationarity. This result may be due to a combination of properties presented by the series such as the high level of linearity (81%), the lowest rate of heteroscedasticity (9%) among the datasets and a considerable amount of autocorrelation, which are all beneficial to ARMA models. These same properties may also have contributed to the somewhat unsatisfactory results of EMD since it tends not to perform well when data presents similar frequencies. Also, since EMD is a flexible nonparametric method, overfitting is a possibility. Furthermore, despite the NN3 time series presenting 70% trend and 57% level stationarity, DT, which removes a deterministic trend by Equation 11, was not among the best methods, suggesting that a polynomial regression as η_t may not provide an explanation of the series behavior. Also, performances from LT10, PCT, LT, and BCT were considered among the best. They were favored by strictly positive data and possible stability in the relative percentage of change in observations. Other interesting results were obtained by the use of MAS, SDIF, and WT transformation methods. Although being able to produce reasonable results, they were still not among the best methods, probably due to the presence of unclear seasonality in the data, in the case of MAS and SDIF, and possible overfitting by the flexible representation of Equation 16, in the case of WT, since the data was mostly linear.

Results on NN5

For an overview of the results of the NN5 dataset, Figure 11c may be observed. The dataset NN5 time series presented the lowest rate of difference stationarity among all datasets. This fact particularly favored the Naive approach, as could be expected, and it was among the best methods. Also, since the NN5 time series presented 71% trend and 50% level stationarity, the deterministic trend removed by DT was suitable for providing stationarity. Other properties presented by the NN5 series are heteroscedasticity in 52% of the series and nonlinearity in 41% of the series (highest rate among the datasets). Transformation methods which are prepared to handle such properties may have been

favored, such as WT. Other examples of such methods, like PCT, LT10, LT, BCT, and EMD could have also been favored, but that was not the case, due to 0 values in 93 of the 111 series of the dataset (impairing the PCT, LT10, LT, and BCT methods) and possibly due to the presence of similar frequencies in the EMD decomposed data, causing worse performances. Conversely, the transformation methods DIF, SDIF, and DIFs produced better predictions. The results of MAS are highlighted being the best method, which indicates the presence of clear long-term behaviors and seasonality in the series, which is to be expected in daily withdrawals at cash machines.

Results on Ipea_M

The Figure 11d is used for discussing the results of the Ipea_M dataset. All series in Ipea_M were actually difference stationary. This property expectedly favored transformation methods like DIF, DIFs, EMD, and WT that remove stochastic trends, apply differencing or use decomposition to derive a stationary time series before prediction. It also impaired the Naive approach since it does not address this property. As it was for the CATS results, SDIF presented performance equal to Naive as no seasonal differencing parameters were estimated by the *auto.arima* function (HYNDMAN; ATHANASOPOULOS, 2013). In spite of having trend and level stationarity on only less than 5% of the time series, DT presented a reasonable performance, probably due to the help of mostly constant variability (only 30% of heteroscedasticity) and high level of linearity (83% linearity in the mean) presented by the time series in the dataset. Results of LT, LT10, and BCT were poor compared to other methods but PCT presented reasonable performance. Although all these latter methods were badly affected by the negative and/or zero values in 26% of the time series in the dataset, PCT benefit from the possible constancy in the percentage of change in the data of economic nature. The MAS method also presented fair performance, favored by the seasonality characteristic of the time series.

Results on Ipea_D

Finally, for the results of the Ipea_D dataset, Figure 11e is presented. Difference stationarity was present in 58% of the series in Ipea_D, impairing the Naive approach which does not address it, and favoring the transformation methods DIF, DIFs, WT, and EMD, which were among the best methods, followed by PCT and DT. The performance of DT was unexpected since there is only 8% of the series in the dataset that present trend or level stationarity, thus not suggesting the need for removal of a deterministic trend for achieving stationarity. The presence of 75% of linear and reasonable 50% of homoscedastic time series may have been advantageous for the performance provided by the use of DT. Non-constant variability in 50% of the series, together with a possible constancy in the percentage of change of the economic data, may have also been in favor of the performance obtained by the use of PCT, that was also among best methods presenting statistically significant prediction improvements over seven other methods. Despite being able to handle variability in the data, much like PCT, the transformation methods LT, LT10, and BCT did not perform well. Since all four transformation methods were not resilient to 20% of the series containing 0 values, a superiority of PCT over this dataset is indicated. Furthermore, it is noted that (for the same reason as presented in previously discussed datasets) SDIF represent the same method as Naive. Other results that are worth mentioning are the ones produced by the use of MAS, which was not among the best methods, probably due to the presence of unclear long-term trends or seasonality in the data.

3.4.4 Summary of results across all datasets

In order to have an overview of the predictions provided by the use of each of the evaluated nonstationary time series transformation methods over all the selected datasets, the plot in Figure 12 was generated which presents the relative number of times that the use of each transformation method led to better predictions across all datasets. In other words, this plot presents the relative number of times each method was not present

within any area demarked by a dashed rectangle in Figure 11, which represent the worst evaluated transformation methods regarding the experiments on each dataset.

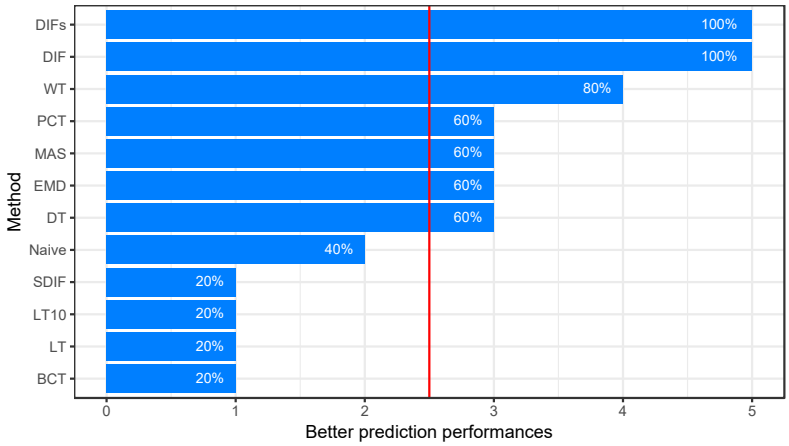


Figure 12 – Plot of the number of times (and percentage) the use of each method provided better predictions over all five evaluated datasets

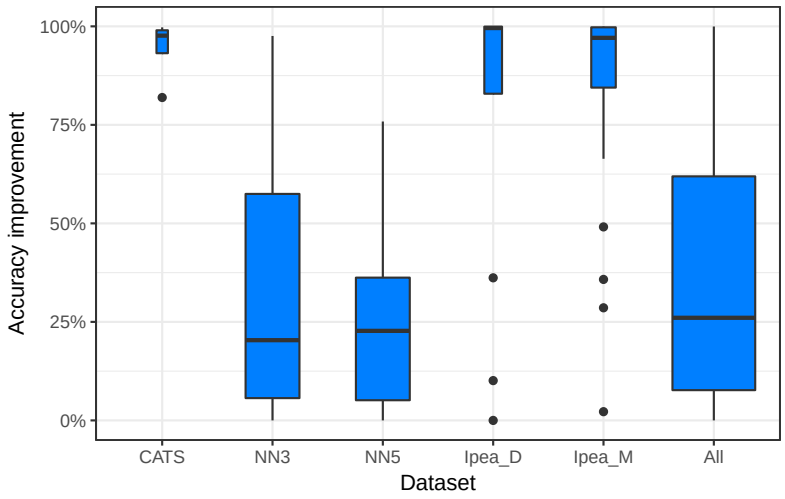
From the analysis of Figure 12, attention is drawn to the results of the methods DIF and DIFs, which were consistently among the best methods. This fact indicates the adequacy of differencing techniques for aiding the prediction of the time series of the selected datasets. The splitting-based time series decomposition methods EMD and WT were also among the best methods throughout the experiments, together with MAS, PCT, and DT. It was also noted that the use of WT provided a more consistent performance than EMD in our experimental analysis. The worst evaluated transformation methods across all datasets (marked before the red vertical line) were expectedly the Naive approach, and the SDIF, LT, LT10, and BCT transformation methods.

3.4.5 Discussion

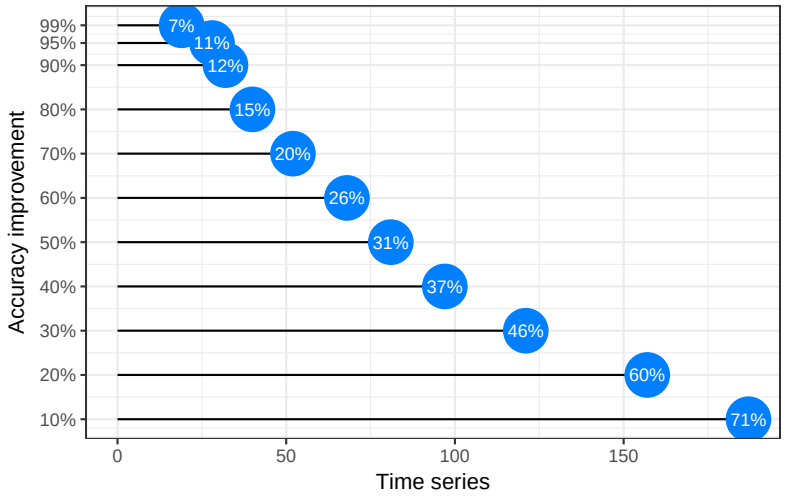
Our experimental results indicate that the effects of each evaluated nonstationary time series transformation methods regarding predictions and coercion of a time series towards stationarity were dependent on the statistical properties and characteristic nature of the data. This observation and the analysis of the information in Figure 12 suggest the importance of a study on the effects and benefits of different nonstationary time series transformation methods to be performed on the dataset at hand before any prediction

tasks.

Moreover, the significance of performing a preliminary data transformation activity may be better visualized in Figure 13. The plots in Figure 13 depict relevant information about observed prediction accuracy improvements (measured by the decrease in MSE error) provided by the best method over the Naive approach for each time series used in this experimental analysis.



(a)



(b)

Figure 13 – Plots of prediction accuracy improvement provided by transformation methods. (a) presents box plots of prediction accuracy improvements provided by the best method for each time series of each dataset. (b) presents the number (and percentage) of time series (over all datasets) for which at least a minimum percentage of prediction accuracy improvement was provided by their best method.

As can be observed in Figure 13a, the datasets for which transformation methods were most effective were CATS, Ipea_D and Ipea_M. This result may be mostly due to the

fact that these are among the datasets which presented the greatest rates of difference stationarity in their time series. This is one of the most challenging forms of nonstationarity and therefore, when not treated, has a significant negative impact on the accuracy of simpler prediction methods such as the ARMA model.

Furthermore, the lower relative accuracy improvements observed for the NN3 and NN5 datasets are probably due to properties of linearity, homoscedasticity, autocorrelation, and also a low rate of difference stationarity (in the case of the NN5 dataset), which are favorable features to the performance of ARMA models. By analyzing the last box plot of Figure 13a, corresponding to all time series of all datasets, it is possible to observe a skew in the results where the best methods lead to prediction accuracy improvements of at most around 30% in half of the time series. This skew is actually to be expected since most data are from the NN3 and NN5 datasets that provided 222 out of the 262 evaluated time series. Nonetheless, the upper quartile and the upper extreme values of the last box plot represent considerably high prediction accuracy improvements obtained by transformation methods over half of the evaluated time series.

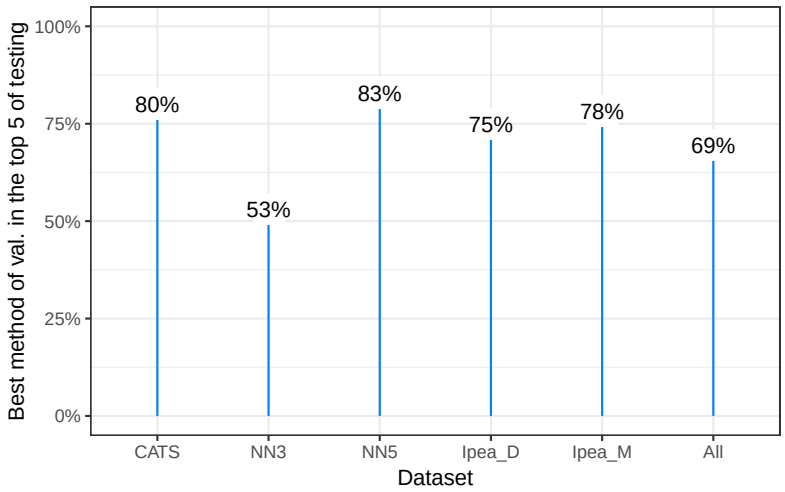
Complementary information may be obtained from Figure 13b that illustrates the number of evaluated time series across all datasets for which at least a minimum percentage of prediction accuracy improvement was achieved by the use of their best method. The plot in Figure 13b confirms and detail the information presented in the last box plot of Figure 13a, showing that for almost 50% of the series the best methods produced prediction accuracy improvements of at least 30%. Moreover, one can observe other significantly higher rates of accuracy improvements reaching at least 70% for 20% of the series, at least 95% for over 10% of the series and a maximum value of at least 99% prediction accuracy improvement for 7% of the time series used in our experiments. The results indicate the potential impact of adequate transformation methods on the problem of accurately predicting times series presenting nonstationary properties.

However, the task of selecting an adequate transformation method for the problem of predicting a particular nonstationary time series is not straightforward. This is confirmed by our previous analysis of the effects of the evaluated nonstationary time series transformation methods on prediction. In particular, our experimental results support a preliminary conclusion that there is no “silver bullet” method for all datasets. Also, the notion of an adequate transformation method is subjective according to each time series prediction application.

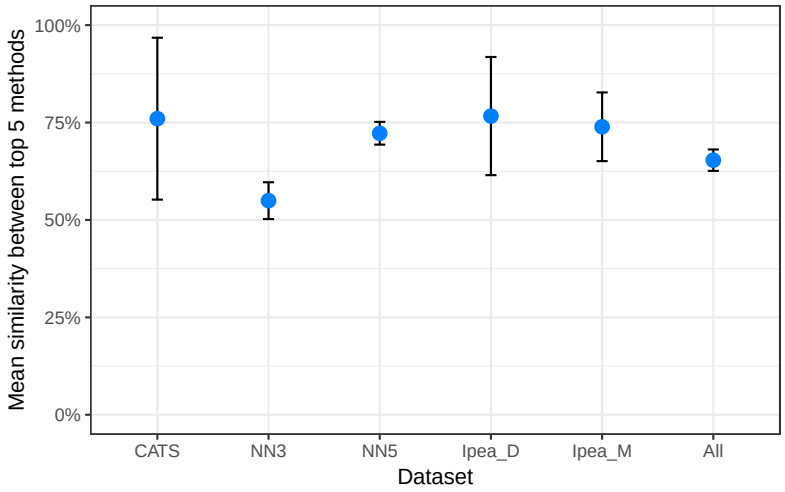
Our experiments pointed to the influence of transformation methods during prediction. Nonetheless, the question remains “is it possible to consistently find appropriate transformation methods for the prediction of a nonstationary time series before testing?”. To address this question, it was applied a validation phase during model training. The validation was introduced to rank candidate methods based on MSE. The suitability of the method selected during validation could than be confirmed during the testing phase of the prediction application (HAN; KAMBER; PEI, 2011) using a similar ranking process. Figure 14 presents plots for comparing rankings of transformation methods based on MSE measures generated from predictions performed during both validation and testing phases. Analogously to our previous discussion, henceforth it is considered the best method the one whose application resulted in the smallest absolute values of MSE prediction errors.

One may consider the rate at which the best method observed during the validation phase is found among the top 5 methods found during the final testing phase. This information is presented in Figure 14a where it is possible to see that, for the majority of the time series adopted in our experiments, the best method found during the validation was in fact confirmed as one of the most adequate (if not the most adequate) transformation methods for aiding the accurate prediction of each time series. Attention is also drawn to the lowest rate of 53% derived from the time series of the NN3 dataset. This rate is mostly due to the relatively small length of the NN3 times series (the smallest among the adopted datasets), a property that is generally challenging for time series modeling. This result helps to suggest that inferences on best methods made on a validation phase are more likely to approximate the truth when sufficient data is available.

The plot in Figure 14b presents another indicator for evaluating the ability of consistently finding adequate transformation methods for aiding the prediction of a time series through a validation phase. It depicts the mean rate of similarity (intersect) between the top 5 best methods ranked during the validation and testing phases. Interpretation of the plot suggests that even in case the best method found in validation phase is not present in the top 5 methods of the testing phase, the top 5 rankings of both phases share on average at least 65% (3) of their transformation methods considering all adopted time series. This observation denotes a relative consistency among the results of both the validation and testing phases. Confidence intervals are also represented. Expectedly, the three wider confidence intervals regarding CATS, lpea_D and lpea_M correspond precisely to the datasets which possess the lowest number of time series (respectively 5, 12 and



(a)



(b)

Figure 14 – Plots for comparing the best methods according to predictions performed during the validation and testing phases. (a) presents the (percentage) number of times the best method found during validation was also present in the top 5 ranked methods found during testing, for each time series of each dataset. (b) presents the mean (percentage) similarity (intersect) between the top 5 ranked methods of both validation and testing phases, for each time series of each dataset.

23 in comparison to 111 for each NN3 and NN5). The analysis of the results illustrated in Figure 14 indicates that it is generally likely to find and select the most adequate nonstationary time series transformation methods for aiding the production of accurate predictions based on the use of a validation phase.

4- Enabling the benchmarking of transformation methods and models

The last chapter presented an application of the benchmarking framework described in Chapter 2 with the goal of comparatively reviewing nonstationary time series transformation methods and their effects on prediction. For this end, a well-established benchmark linear model was adopted for predicting the selected time series datasets. This chapter gives use case examples designed to indicate the potential of the benchmarking framework for general nonstationary time series prediction applications with MLM. Applicability encompasses the selection of hyperparameters and the choice of adequate transformation methods and prediction models for a particular nonstationary time series. Henceforth this work focus on benchmarking transformation methods and MLM combinations in order to discuss their adequacy to the time series of the CATS dataset. The CATS dataset is selected for the considerably high nonstationarity rate of its time series as described in Section 3.1. The code for reproducing the use case examples presented in this chapter and their respective results are made available [1] .

4.1- Use case 1: choice of hyperparameters

The first use case example of the usability of the framework developed in this research encompasses the problem of hyperparameter selection for MLM. With this purpose, the benchmarking comparison process described in Section 2.1 was instantiated for each time series of the CATS dataset. Several hyperparameter candidates were benchmarked based on their ability to produce models that yield the most accurate predictions.

The MLP network model was adopted for training and prediction of the next 20 time series observations. MLPs present one of the most common network architectures

[1] https://github.com/RebeccaSalles/TSPred/wiki/tspred-use-cases

and they were trained by standard error backpropagation (BERGMEIR; BENÍTEZ, 2012). Since the focus of this use case example lies on the selection of model parameters, no preprocessing methods were applied to the selected time series except the subsetting into training and evaluation datasets and the normalization with MM during training and prediction. Analogous to the experiments in Chapter 3, prediction accuracy evaluation was based on MSE metrics.

Regarding the model parameter options set to be benchmarked in this use case example, it is worth mentioning that the number of hidden layers of the network is defined as either one or two and the number of units in these layers varies within the range $[2, 20]$ for the first input layer and $[0, 20]$ for the second. The learning rate of the backpropagation learning function specifies the gradient descent step width and it is usually set to a value between 0.1 and 1. Thus the learning rate was set to vary between these values with a 0.1 step increment. Finally, the maximum number of iterations of the MLP learning algorithm was set as either 1000, 5000 or 10000. The described parameter options generate a number of $12,540$ different combinations for each of the 5 CATS times series.

Algorithm 1 presents a summary of the experimental methodology applied in this use case example for each time series of the CATS dataset. The use case initially receives three parameters: (i) a time series X; (ii) a number m of observations to be predicted; and (iii) a set of parameter combinations to serve as hyperparameter candidates, A.

Algorithm 1 – Experimental methodology of use case 1

Input: $X =$ experimental time series data;
$m =$ number of observations to be predicted;
$A =$ set of parameter combinations
Output: $\hat{\alpha} =$ selected hyperparameter candidate

```
1  begin
2      X_eval ← DataSampling(X, m);
3      X_train ← X − X_eval;
4      foreach parameter combination α in A do
5          μ_α ← TrainMLP(X_train, α);
6          ρ_α ← Predict(μ_α, X_train, m);
7          ε_α ← Evaluate(ρ_α, X_eval);
8          ε_A ← Append(ε_A, ε_α);
9      R ← Benchmark(ε_A, A);
10     α̂ ← Top1(R)
```

In lines 2 and 3, X is divided into two disjoint subsets, X_{eval} and X_{train}, corresponding to the previously described evaluation and training sets, respectively. The former

subset needs to have no more than m observations to enable the subsequent calculation of the prediction errors, therefore it is assigned the length of X_{eval} equal to m, and for X_{train} it is assigned all the remaining observations contained in X, that is, the length of X minus the m observations.

From lines 5 to 7, the algorithm performs training, predicting and error evaluation. These tasks are iteratively performed based on each hyperparameter candidate α contained in A. The first executed task, in line 5, is the training of the MLP model based on α and the data within X_{train}, which gives us the trained MLP model μ_α. μ_α is then used to predict the m next observations of the time series present in X_{train}. The set of predictions are then represented by ρ_α. This task can be observed in line 6 of Algorithm 1. Line 7 computes the prediction error of the trained model using X_{eval}. Let ϵ_α be the prediction error produced by μ_α. Line 8, appends ϵ_α to ϵ_A, the set of prediction errors produced based on each α in A.

In line 9 the hyperparameter candidates in A are benchmarked based on the ranking of their respective prediction errors contained in ϵ_A generating a performance ranking referenced as R. Finally, the algorithm selects $\hat{\alpha}$ as hyperparameter for the MLP model of the time series in X by taking the best-ranked candidate in R. Henceforth, this work refers as *Product 1* the collection of the hyperparameters, $\hat{\alpha}$, selected for each time series of CATS. The *Product 1* produced by this use case example is presented in Table 5. V1 to V5 represent the five consecutive sequences of known values of the CATS series that were predicted separately.

Table 5 – Hyperparameters selected for MLP modeling of each time series of the CATS dataset. This collection of hyperparameters compose the *Product 1* produced by this use case example.

Time series	Number of hidden layers	Units in layer 1	Units in layer 2	Learning rate	Maximum iterations
V1	2	17	11	0.1	10000
V2	1	20	NA	0.3	10000
V3	1	20	NA	0.8	1000
V4	2	6	10	1	10000
V5	2	17	0	0.6	5000

The resulting prediction errors accumulated in ϵ_A for each CATS time series are presented in the boxplot of Figure 15. A clear discrepancy can be seen regarding the prediction results for the time series V4 corresponding to the fourth sequence of known

values (3001-3980) of the CATS time series. The prediction errors computed for *V4* are considerably higher than for all other time series. It is possible to have an intuition of the cause of this result by observing the V4 time series in Figure 16. V4 presents highly nonstationary behavior and a steep positive trend in its final portion, which is also abruptly interrupted in the next observations of the CATS series. In fact, the demand for addressing the property of nonstationarity for improving prediction accuracy is confirmed for all time series of CATS. This demand is to be expected given the inherent nonstationary nature of the CATS dataset. Moreover, it is also indicated by the red solid marks in Figure 16, which represent the benchmark ARIMA prediction errors for the time series. Since the benchmark model produced similar or even smaller prediction errors, a refinement in the prediction setup is generally recommended.

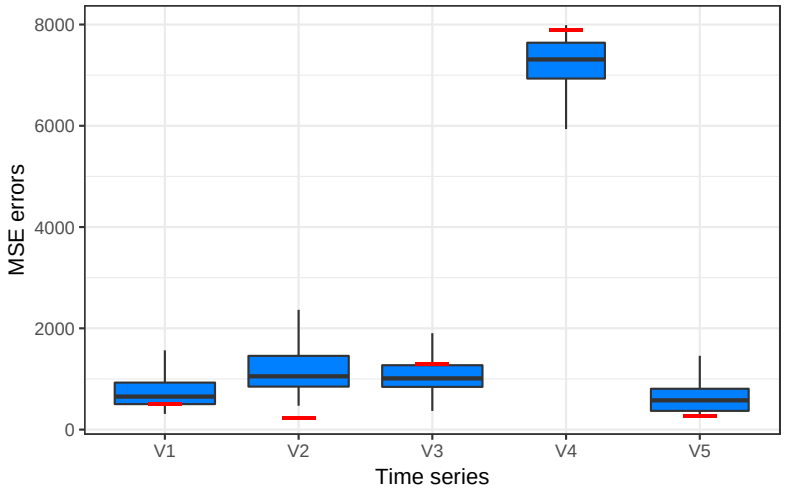


Figure 15 – Boxplot of MSE prediction errors generated in the use case 1. The red solid marks represent prediction errors produced by a benchmark ARIMA model.

4.2- Use case 2: choice of transformation methods

The second use case example of the usability of the developed framework encompasses the problem of selecting appropriate transformation methods for a particular nonstationary time series. For that goal, analogous to the use case 1, a benchmarking comparison process was instantiated for the time series of the CATS dataset. However, this use case benchmarks several time series transformation method candidates based

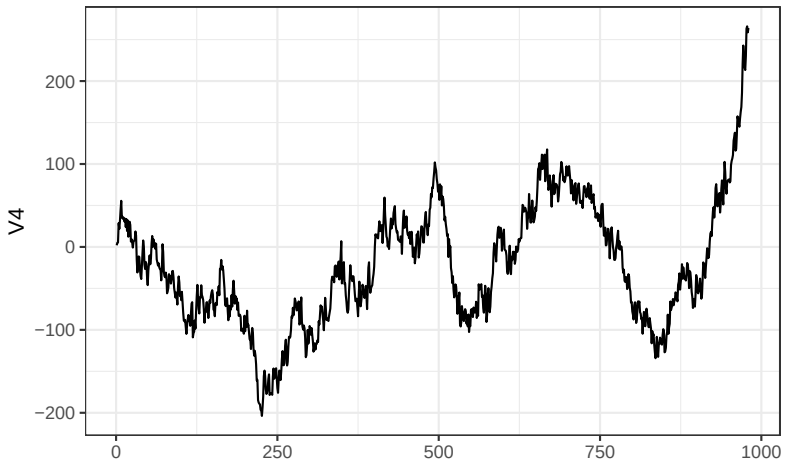


Figure 16 – The forth sequence of known values (3001-3980) of the CATS time series referenced as V4

on their ability to produce transformed time series that can be most accurately predicted.

Similarly to use case 1, the adopted model for training and prediction was the MLP network model, and the MSE metrics served as a base for prediction accuracy evaluation. It is important to mention that for practical applications a hyperparameter selection process (as the one performed in use case 1) should be performed for each of the time series transformed by the candidate transformation methods being benchmarked. Nonetheless, for the purposes of this example, the hyperparameters selected in use case example 1, *Product 1* (Table 5), were adopted for model training.

The transformation method options set to be benchmarked in this use case example were selected based on the top 5 methods depicted in Figure 12, namely the DIF, WT, PCT, MAS, and EMD. The Naive method, that is, when no transformation methods are applied, is also benchmarked against the listed top 5. These methods were also combined with two options of normalization methods applied during the MLP network training and prediction, namely the MM and the AN. The described transformation/normalization options generate a number of 12 different combinations for each of the 5 CATS times series.

Algorithm 2 presents a summary of the experimental methodology applied in this use case example for each time series of the CATS dataset. The use case initially receives five parameters: (i) a time series X; (ii) a number m of observations to be predicted; (iii) model hyperparameters $\hat{\beta}$; (iv) a set of transformation method candidates to be benchmarked, B_1; and (v) a set of normalization method candidates, B_2, to be benchmarked in combination with B_1.

Algorithm 2 – Experimental methodology of use case 2

Input: $X =$ experimental time series data;
$m =$ number of observations to be predicted;
$\hat{\alpha} =$ model hyperparameters;
$B_1 =$ set of transformation method candidates;
$B_2 =$ set of normalization method candidates
Output: $\hat{\beta} =$ selected transformation-normalization combination

```
1  begin
2      X_eval ← DataSampling(X, m);
3      X_train ← X − X_eval;
4      foreach transformation method β_1 in B_1 do
5          foreach normalization method β_2 in B_2 do
6              Ẋ_train ← Preprocess(β_1, X_train);
7              μ̇_β ← TrainMLP(Ẋ_train, α̂, β_2);
8              ρ̇_β ← Predict(μ̇_β, Ẋ_train, m);
9              ρ_β ← Postprocess(β_1, ρ̇_β);
10             ε_β ← Evaluate(ρ_β, X_eval);
11             ε_B ← Append(ε_B, ε_β);
12     R ← Benchmark(ε_B, B_1, B_2);
13     β̂ ← Top1(R)
```

Analogously, lines 2 and 3, divide X into the evaluation and training sets, X_{eval} and X_{train}, respectively. From lines 6 to 10, the algorithm performs preprocessing, training, predicting, postprocessing and error evaluation. These tasks are iteratively performed based on each transformation β_1 and each normalization β_2 method candidate contained in B_1 and B_2, respectively. In line 6, the X_{train} is transformed using the method β_1, resulting in $\dot{X}_{train}$. The next task is the training of the MLP model based on the transformed data within $\dot{X}_{train}$, $\hat{\alpha}$, and the data normalization method β_2, which gives us the trained MLP model $\dot{\mu_\beta}$. $\dot{\mu_\beta}$ is then used to predict the m next observations of the transformed time series present in $\dot{X}_{train}$. The set of predictions of the transformed time series are then represented by $\dot{\rho_\beta}$. The predictions for the original time series ρ_β are obtained by reverse transforming $\dot{\rho_\beta}$ in line 9. Line 10 computes the prediction error ϵ_β. Line 11, appends ϵ_β to ϵ_B, the set of prediction errors produced based on each β_1 and β_2 combinations.

In line 12 the transformation-normalization combination candidates are benchmarked based on the ranking of their respective prediction errors contained in ϵ_B generating R. Finally, the algorithm selects $\hat{\beta}$ as the most appropriate transformation-normalization combination for the MLP prediction of the time series in X by taking the best-ranked candidate in R. Henceforth, this work refers as *Product 2* the collection of the transformation-

normalization combinations, $\hat{\beta}$, selected for each time series of CATS. The *Product 2* produced by this use case example is presented in Table 6.

Table 6 – Transformation-normalization combinations selected for MLP prediction of each time series of the CATS dataset. This collection of preprocessing method combinations compose the *Product 2* produced by this use case example.

Time series	Transformation method	Normalization method
V1	DIF	AN
V2	Naive	MM
V3	PCT	AN
V4	WT	AN
V5	EMD	MM

The mean and standard deviation of the prediction errors accumulated in ϵ_B for each CATS time series are presented in the plot of Figure 17. The results were aggregated by each benchmarked transformation method (β_1), listed in the x-axis, and each benchmarked normalization method (β_2), referenced by the colored points/lines. By analyzing the plot, it is clear to see that the AN normalization method helped provide more accurate predictions than MM. This result is justified given that the AN normalization method was designed to address properties such as the nonstationarity in the time series. It is also possible to observe that the standard deviations were considerably smaller for the transformation methods based in time series decomposition, namely, the EMD and the WT. Consequently, they were able to inspire a higher confidence in their respective prediction results despite the model hyperparameters being tuned to the Naive method (in use case 1), which favored its relative prediction performance in this use case example.

It is important to remark that although this use case example focused on benchmarking predictions of time series preprocessed by only one transformation method at a time, this is not the only possible configuration. The developed framework also allows the preprocessing of time series by consecutively applying a combination of a number (n) of transformation methods defined by the user. Furthermore, the benchmarking process presented in this use case example can be adapted by adding iterations over each transformation method ($\beta_1, \ldots, \beta_n$) of the combination.

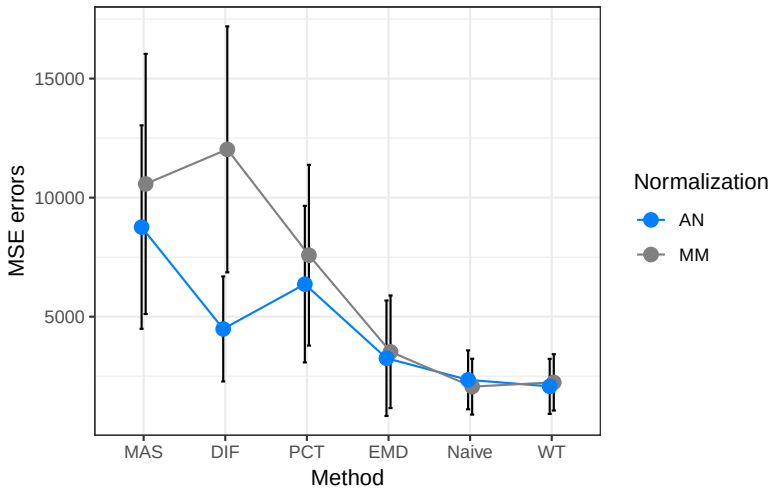


Figure 17 – Mean and standard deviation of MSE prediction errors generated for each CATS time series in the use case 2. Results are aggregated by the applied transformation (x-axis) and normalization methods (color).

4.3- Use case 3: choice of models

The third and last use case example of the usability of the developed framework encompasses the problem of selecting an adequate machine learning model for a particular nonstationary time series prediction application. With this intent, a benchmarking comparison process that is analogous to both previous use cases (1 and 2) was instantiated for the same time series dataset. In this case, the focus of the use case shifted to benchmarking time series MLM candidates based on their ability to produce the most accurate predictions.

It is reiterated that for practical applications a hyperparameter selection process (use case 1) and a transformation method selection process (use case 2) should be performed for each candidate MLM being benchmarked. However, for the purposes of this example, model parameters were arbitrarily selected and the combinations of transformation/normalization methods selected in use case example 2, *Product 2* (Table 6), were adopted for data preprocessing. The MSE was again the chosen metric for prediction accuracy evaluation.

The MLM options set to be benchmarked in this use case example were the NNET, RFrst, RBF, SVM, MLP and ELM. The linear ARIMA model is also benchmarked against the listed MLM serving as a well-established reference. It is noted that for the MLM based

on neural networks, namely the NNET, RBF, MLP and ELM, the number of hidden layers is set to one and its neuron units are set based on partial autocorrelation tests. The number of neurons set for each CATS time series is presented in Table 7. The size of the sliding windows was set by incrementing the values in Table 7 by one unit. Also, no data normalization methods were applied for ARIMA predictions. Given that ARIMA does not receive sliding windows data as input, the normalization methods implemented for such data could not be applied. The listed models provide a number of 7 different options for each of the 5 CATS times series.

Table 7 – The number of neuron units in the hidden layer of the benchmarked MLM based on neural networks. Values were set based on partial autocorrelation tests performed for each CATS time series.

Time series	Neuron units
V1	17
V2	17
V3	28
V4	17
V5	17

Algorithm 3 presents a summary of the experimental methodology applied in this use case example for each time series of the CATS dataset. The use case initially receives six parameters: (i) a time series X; (ii) a number m of observations to be predicted; (iii) a transformation method $\hat{\beta}_1$; (iv) a normalization method $\hat{\beta}_2$ to be applied in combination with $\hat{\beta}_1$; (v) a set Γ of model candidates to be benchmarked; and (vi) a set of parameters, A_γ, respective to the models in Γ.

Analogous to the use cases 1 and 2, after dividing X into X_{eval} and X_{train} (lines 2 and 3), the algorithm performs the tasks of preprocessing, training, predicting, post-processing and error evaluation, which are iteratively performed based on each model candidate γ contained in Γ. In line 5, X_{train} is transformed by the method $\hat{\beta}_1$. In the next task, the γ model is trained based on the transformed data $\dot{X}_{train}$, the parameters α_γ, and the normalization method $\hat{\beta}_2$, resulting in the trained model $\dot{\mu_\gamma}$. $\dot{\mu_\gamma}$ is used to predict the m next observations of the transformed time series, represented by $\dot{\rho_\gamma}$. The predictions for the original time series ρ_γ are obtained by reverse transforming. Line 9 computes the prediction error ϵ_γ. Line 10, appends ϵ_γ to ϵ_Γ, the set of prediction errors produced based on each model γ.

In line 11 the model candidates are benchmarked based on the ranking of their

Algorithm 3 – Experimental methodology of use case 3

Input: $X =$ experimental time series data;
$m =$ number of observations to be predicted;
$\hat{\beta}_1 =$ transformation method;
$\hat{\beta}_2 =$ normalization method;
$\Gamma =$ set of model candidates;
$A_\gamma =$ set of parameters for the models of Γ;
Output: $\hat{\gamma} =$ selected model

begin
  $X_{eval} \leftarrow$ DataSampling(X, m);
  $X_{train} \leftarrow X - X_{eval}$;
  foreach model γ in Γ and respective α_γ in A_γ **do**
    $\dot{X}_{train} \leftarrow$ Preprocess$(\hat{\beta}_1, X_{train})$;
    $\dot{\mu_\gamma} \leftarrow$ Train$(\gamma, \dot{X}_{train}, \alpha_\gamma, \hat{\beta}_2)$;
    $\dot{\rho_\gamma} \leftarrow$ Predict$(\dot{\mu_\gamma}, \dot{X}_{train}, m)$;
    $\rho_\gamma \leftarrow$ Postprocess$(\hat{\beta}_1, \dot{\rho_\gamma})$;
    $\epsilon_\gamma \leftarrow$ Evaluate(ρ_γ, X_{eval});
    $\epsilon_\Gamma \leftarrow$ Append$(\epsilon_\Gamma, \epsilon_\gamma)$;
  $R \leftarrow$ Benchmark$(\epsilon_\Gamma, \Gamma)$;
  $\hat{\gamma} \leftarrow$ Top1(R)

respective prediction errors contained in ϵ_Γ generating R. Finally, the algorithm selects $\hat{\gamma}$ as the most adequate model for the prediction of X by taking the best-ranked candidate in R. Henceforth, this work refers as *Product 3* the collection of models, $\hat{\gamma}$, selected for each time series of CATS. The *Product 3* produced by this use case example is presented in Table 8.

Table 8 – Models selected for prediction of each time series of the CATS dataset. This collection of models compose the *Product 3* produced by this use case example.

Time series	Model
V1	ELM
V2	ARIMA
V3	ELM
V4	ARIMA
V5	SVM

A Taylor diagram of the predictions obtained for the first sequence of known values (1-980) of the CATS series, V1, is presented in Figure 18. As described in Section 3.4.1, the closer to the point ”observed”, the better the prediction performance of the benchmarked model. In this case, the diagram indicates that the most accurate predictions were produced by the models RFrst, ELM, SVM, MLP, and RBF, respectively. The use

of the benchmark model ARIMA resulted in considerably higher prediction errors, as it should be expected. Nonetheless, NNET produced the worst prediction results, despite presenting a higher correlation (0.85) to the evaluation dataset (X_{eval}) than the results of SVM, MLP and RBF, for example. The ranked MSE computed for the prediction results in the diagram of Figure 18 is presented in Table 9. The ranking of models remains practically the same (except for the RFrst) and the ELM is selected as the best-placed candidate model. Again the benchmark ARIMA model was able to indicate a demand for refining data preprocessing methods and/or parameters of models such as the NNET.

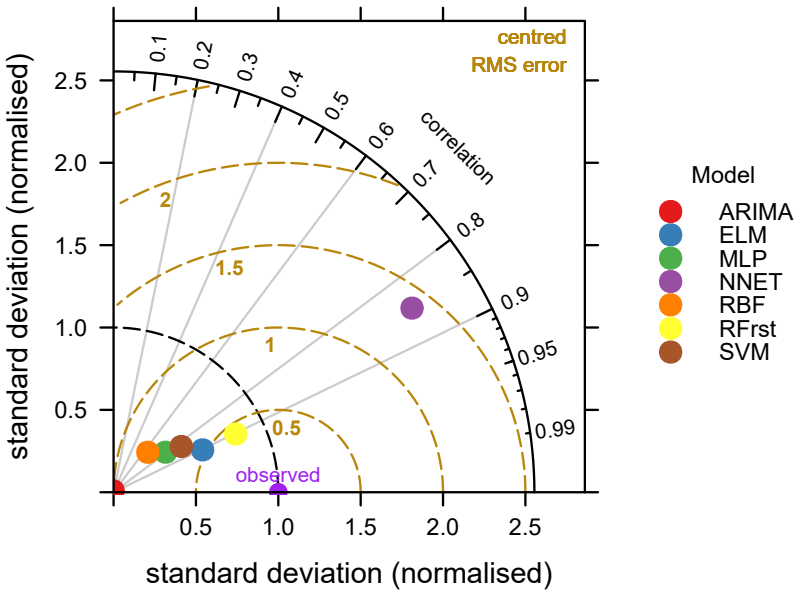


Figure 18 – Taylor diagram of the predictions obtained for V1 in the use case 3

Table 9 – MSE prediction errors obtained for V1 in the use case 3

Model	MSE
ELM	120.0758
SVM	197.2222
MLP	242.8818
RBF	308.9310
RFrst	408.7529
ARIMA	583.7616
NNET	2941.8614

4.4- Discussion

The use case examples presented in this chapter indicate the potential of the developed framework for benchmarking nonstationary time series prediction. It enables a comparative analysis of candidate models, preprocessing methods, and parameters for selecting the most adequate setup for a particular nonstationary time series prediction application.

The described use case examples of the previous sections are in fact complementary. They should be combined in order to obtain the most accurate predictions in real-world practical time series applications. It is also recommended that the definition of candidate models, preprocessing methods, and their parameters for the performed benchmarking process should be refined based on application expert background.

Assume the same benchmarking candidate options are used in a composite of the previously described use case examples (1, 2 and 3). The number of possible candidate setup combinations reaches a number of $12,540 \times 12 \times 7 = 1,053,360$ for each of the 5 CATS times series. In total, there are $5,266,800$ different time series prediction setups to be performed and benchmarked, which is a very high and computationally challenging number.

Performing an entire prediction process (Section 2.1) with an MLM model, such as the MLP, takes in average 30 seconds. This value is valid for a computer with an 8th generation Intel i5 with 8Gb of RAM memory running Windows 10. Therefore, the monolithic computation of predictions for all $5,266,800$ setup combinations may take up to $158,004,000$ seconds, or 5 years, to be complete. In this case, there is a clear demand for adopting parallel and/or distributed computing and optimization in order to make such benchmarking process feasible. It is also important to study a methodology for creating heuristics and pruning the options of candidate models, preprocessing methods, and parameters based on the particular time series data and prediction application at hand.

Final considerations

This work focus on the study of univariate nonstationary time series prediction and the benchmarking of preprocessing and modeling options for time series applications that have nonstationarity as an inherent property. It is presented a review of nonstationary time series transformation methods for time series prediction. A categorization of such transformation methods was described together with a timeline obtained through a systematic mapping study. Moreover, it was developed a systematic framework for benchmarking transformation methods and models for nonstationary time series prediction. This framework was implemented and encapsulated within the *TSPred* R-package (SALLES; OGASAWARA, 2018), which is publicly available.

The developed benchmarking framework was adopted for devising a comparative experimental analysis and discussion of the effects of some of the reviewed transformation methods on the problem of time series prediction. With this intent, eleven methods of the most commonly used in practical applications were selected and benchmarked. The aim of this experimental analysis is contributing to the process of evaluation, selection, and application of nonstationary time series transformation methods.

An overview of the effects of the evaluated methods regarding predictions and stationarity was produced based on our experimental results. Although it was possible to note a somewhat consistency in the results of the evaluated transformation methods, there was no uniquely best method across all datasets, and the nature and statistical properties of the time series were especially relevant to the results.

Nonetheless, it was possible to observe better predictions when transformation methods based on differencing and moving average smoothing were applied before the prediction of the time series of the selected datasets. Transformation methods that perform time series decomposition, which have been an object of increasing attention, were also among the best methods. Among the worst methods was, as expected, the naive one, where no data transformation is performed before prediction. Particularly, this approach provided predictions with significantly lower accuracy when compared to the case in which nonstationarity was treated.

Additionally, results indicate that the use of a validation phase for exploring different

transformation methods generally leads to the selection of one of the most appropriate for obtaining accurate time series predictions. Our experimental results suggest as future trends (i) the increase in the importance of the process of data transformation for the problem of accurate prediction of nonstationary time series and (ii) the need for studying and evaluating suitable methods to perform this activity according to the dataset at hand.

In this context, the potential of the developed framework for enabling the benchmarking of data transformation methods and prediction models for a particular nonstationary time series application was indicated. With this goal, this work presents use case examples of the framework usability encompassing the selection of hyperparameters, and the choice of adequate transformation methods and machine learning prediction models. For example purposes, the use cases benchmark the top 5 evaluated transformation methods and six different MLM for prediction of 5 selected nonstationary time series. The benchmark linear ARIMA model is also adopted to indicate demands for the refining of preprocessing methods and model parameters. Results are analyzed and the general methodologies for benchmarking and selecting adequate prediction setups for a particular nonstationary time series are described.

Scientific production

The study conducted around the topic of nonstationary time series prediction resulted in the publication of four main scientific research products (SALLES et al., 2016, 2017; SALLES; OGASAWARA, 2018; SALLES et al., 2019). The paper of Salles et al. (2016) was published in the Ecological Informatics journal. It performs an experimental analysis of time series predictions based on nonstationary sensor data of the sea surface temperature (SST) of the tropical Atlantic ocean. The data is collected by the Prediction and Research Moored Array in the Tropical Atlantic (PIRATA) project (GOOS-BRASIL, 2015). The paper focused on evaluating the influence of temporal aggregation in predicting step-ahead SST considering different prediction horizons and different sizes for training datasets. Results point out scenarios indicating whether or not temporal aggregated SST time series may be beneficial for prediction. The improvement of SST prediction is important for aiding the identification of extreme environmental events such as droughts.

The paper of Salles et al. (2017) was published in the proceedings of the International Joint Conference on Neural Networks (IJCNN) held at Anchorage, Alaska, USA. It presents a framework for systematic benchmarking MLM against well-known LM, namely Polynomial Regression and models in the ARIMA family, used as benchmark models for univariate time series prediction. This implementation was evaluated using a wide number of datasets from past prediction competitions. The results showed that fittest LM provided by the framework are adequate benchmark models for performance assessment of univariate time series predictions.

The scientific research content presented in this text was also published and is currently available. The framework described in Chapter 2 automatizes the time series prediction process including the tasks of data preprocessing, modeling, prediction, data postprocessing and the evaluation of prediction quality. Several methods related to each of these tasks are implemented with the incorporation of automatic choice of parameters. Moreover, the structure of the framework was designed for supporting the custom user implementation of methods in a straightforward manner.

The framework offers tools for benchmarking different preprocessing methods and models for the prediction of nonstationary time series of a particular application. Being widely available within the *TSPred* R-package (SALLES; OGASAWARA, 2018) in CRAN, the potential for application of this framework encompasses the areas of statistical sciences, natural sciences, socioeconomics, finance, industry and business. By the end of 2018, the previous version 4.0 of *TSPred* had an average of 680 downloads per month worldwide. Given its comprehensiveness and practical use, it is expected that the advent of its new version 5.0, incorporating the described framework, brings a significantly higher utilization rate.

Finally, the review and experimental analysis of nonstationary time series transformation methods presented in this text (Chapters 1 and 3) were published in the journal Knowledge-Based Systems (SALLES et al., 2019). The paper focused on contributing to the choice of a transformation that is appropriate to the adopted data model and to the problem at hand. It provides a background on the subject of nonstationary time series transformation methods and a discussion on the scenarios they could be most beneficial to the problem of time series prediction.

Future work

Given the expanding importance of the subject of time series prediction and the pervasive presence of nonstationarity in most real-world time series applications, there is a wide scope for future research endeavors. Among them, it is noted the expansion of the range of implemented preprocessing methods, MLM, and evaluation metrics of the framework developed in the *TSPred* R-package. An example of useful addition to the range of implemented preprocessing methods would be some state-of-the-art imputation techniques for treating missing values among nonstationary time series data (SALLES et al., 2015). There is also an increasingly important demand for deriving R-packages complementary to *TSPred* to provide support for multivariate and spatio-temporal series data, which are also commonly found in practical real-world time series applications.

As discussed in Section 4.4, the number of possible combinations of candidate models, preprocessing methods, and hyperparameters that need to be benchmarked in order to select the most adequate prediction setup for a particular nonstationary time series application can become very high and computationally challenging. For enabling the automatic optimization of the task of selecting a most adequate time series prediction setup, there is a demand for creating an algebraic approach for the execution of the workflow defined by the prediction process implemented in *TSPred* (OGASAWARA et al., 2011). Also, for pruning the number of candidate setup combinations, it would be beneficial to create heuristics based on machine learning analysis and according to the inherent properties of a time series. In this context, it is also possible to make the time series prediction process more systematic with the use of Automated Machine Learning (AutoML) techniques. They help provide automatic recommendations for prediction methods and parameters while reducing the need for human interaction.

Moreover, in order to improve the computational feasibility of the task of selecting a most adequate time series prediction setup for a particular nonstationary time series, it is crucial to integrate parallel and distributed computing technologies such as Spark (ZAHARIA et al., 2016; VENKATARAMAN et al., 2016) into the framework implemented in *TSPred*. Another helpful approach is to describe the workflow implemented in *TSPred* and its respective objects using the Predictive Model Markup Language (PMML) (ALEX GUAZZELLI; WILLIAMS, 2009). The use of PMML contributes for the integration of

TSPred with other modeling and prediction tools currently available in the literature.

A short-term plan of future publications includes a paper on the *TSPred* package to be submitted to the Journal of Statistical Software. An application of the package for predicting worldwide fertilizers consumption is also planned. Finally, efforts are being applied into generating and publishing a novel LM called Autoregressive Adaptive Integrated Moving Average (ARAIMA), which combines the ARIMA model with features from the AN method.

References

ABRAHAM, B.; BALAKRISHNA, N. Ch. 29. Time series in industry and business. **Handbook of Statistics**, v. 22, p. 1055–1106, 2003.

AKPINAR, M.; YUMUSAK, N. Year ahead demand forecast of city natural gas using seasonal time series methods. **Energies**, v. 9, n. 9, 2016.

ALDRICH, Eric. **wavelets: A package of functions for computing wavelet filters, wavelet transforms and multiresolution analyses**. [S.l.], Dec. 2013.

ALEX GUAZZELLI, Wen-Ching L.; WILLIAMS, Graham. PMML: An Open Standard for Sharing Models. **The R Journal**, v. 1, n. 1, p. 60–65, May 2009. Available from: ¡http://journal.r-project.org/archive/2009-1/RJournal_2009-1_Guazzelli+et+al.pdf¿.

AN, X. et al. Wind farm power prediction based on wavelet decomposition and chaotic time series. **Expert Systems with Applications**, v. 38, n. 9, p. 11280–11285, 2011.

ATTO, A. M. a; BERTHOUMIEU, Y. b. Wavelet packets of nonstationary random processes: Contributing factors for stationarity and decorrelation. **IEEE Transactions on Information Theory**, v. 58, n. 1, p. 317–330, 2012.

BAILLIE, R. T. Long memory processes and fractional integration in econometrics. **Journal of Econometrics**, v. 73, n. 1, p. 5–59, 1996.

BARBA, L.; RODRÍGUEZ, N. A Novel Multilevel-SVD Method to Improve Multistep Ahead Forecasting in Traffic Accidents Domain. **Computational Intelligence and Neuroscience**, v. 2017, 2017. DOI: 10.1155/2017/7951395.

BARTOŃ, Kamil. **MuMIn: Multi-Model Inference**. [S.l.: s.n.], July 2018. Available from: ¡https://CRAN.R-project.org/package=MuMIn¿. Visited on: 14 Jan. 2019.

BERGMEIR, Christoph; BENÍTEZ, José M. Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS. **Journal of Statistical Software**, v. 46, n. 7, p. 1–26, 2012. Available from: ¡http://www.jstatsoft.org/v46/i07/¿.

BHATTACHARYA, T. K.; BASU, T. K. Medium range forecasting of monthly energy demand via walsh transform. **International Journal of Systems Science**, v. 23, n. 3, p. 297–310, 1992.

BISCHL, Bernd et al. **mlr: Machine Learning in R**. [S.l.: s.n.], Aug. 2016. Visited on: 29 Oct. 2016.

BOKDE, N.; FEIJÓO, A.; KULAT, K. Analysis of differencing and decomposition preprocessing methods for wind speed prediction. **Applied Soft Computing Journal**, v. 71, p. 926–938, 2018. DOI: 10.1016/j.asoc.2018.07.041.

BOX, George E. P.; JENKINS, Gwilym M.; REINSEL, Gregory C. **Time Series Analysis: Forecasting and Control**. 4. ed. Hoboken, N.J: Wiley, June 2008. ISBN 978-0-470-27284-8.

BRANDAO, R. M. a c; NOVA, A. M. O. P. b d. Analysis of nonstationary stochastic simulations using classical time-series models. **ACM Transactions on Modeling and Computer Simulation**, v. 19, n. 2, 2009.

BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct. 2001. ISSN 1573-0565. DOI: 10.1023/A:1010933404324. Available from: ¡https://doi.org/10.1023/A:1010933404324¿.

BROCKWELL, A. E. Likelihood-based analysis of a class of generalized long-memory time series models. **Journal of Time Series Analysis**, v. 28, n. 3, p. 386–407, 2007.

BUZA, Krisztian. Time Series Classification and Its Applications. In: PROCEEDINGS of the 8th International Conference on Web Intelligence, Mining and Semantics. Novi Sad, Serbia: ACM, 2018. (WIMS '18), 4:1–4:4. ISBN 978-1-4503-5489-9. DOI: 10.1145/3227609.3227690.

CAPORALE, G. M. a; GIL-ALANA, L. A. b. Nonlinearities and fractional integration in the US unemployment rate. **Oxford Bulletin of Economics and Statistics**, v. 69, n. 4, p. 521–544, 2007.

CAPORALE, G.M.; SKARE, M. Long memory in UK real gdp, 1851-2013: An arfima figarch analysis. **Transformations in Business and Economics**, v. 17, n. 1, p. 255–268, 2018.

CARMONA, René. **Statistical Analysis of Financial Data in R**. 2. ed. Heidelberg: Springer, Dec. 2013. ISBN 978-1-4614-8787-6.

CAVALIERE, G.a; TAYLOR, A.M.R.b. Heteroskedastic time series with a unit root. **Econometric Theory**, v. 25, n. 5, p. 1228–1276, 2009.

CHENG, C. et al. Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. **IIE Transactions (Institute of Industrial Engineers)**, v. 47, n. 10, p. 1053–1071, 2015.

CHILÈS, J.-P.; DELFINER, P. **Geostatistics: Modeling Spatial Uncertainty: Second Edition**. [S.l.: s.n.], 2012.

CHIROMA, H. a h et al. A Review on Artificial Intelligence Methodologies for the Forecasting of Crude Oil Price. **Intelligent Automation and Soft Computing**, v. 22, n. 3, p. 449–462, 2016.

CLAVERIA, O. a; TORRA, S. b. Forecasting tourism demand to Catalonia: Neural networks vs. time series models. **Economic Modelling**, v. 36, p. 220–228, 2014.

CLEMENTS, Michael P.; HENDRY, David F. (Eds.). **A Companion to Economic Forecasting**. 1 edition. Malden, Mass.: Wiley-Blackwell, Mar. 2005. ISBN 978-1-4051-2623-6.

CONEJO, A. J. et al. Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. **IEEE Transactions on Power Systems**, v. 20, n. 2, p. 1035–1042, May 2005. ISSN 0885-8950.

CORONA, F.; GONZÁLEZ-FARÍAS, G.; ORRACA, P. A dynamic factor model for the Mexican economy: are common trends useful when predicting economic activity? **Latin American Economic Review**, v. 26, n. 1, 2017. DOI: 10.1007/s40503-017-0044-7.

CORONA, F.; PONCELA, P.; RUIZ, E. Determining the number of factors after stationary univariate transformations. **Empirical Economics**, v. 53, n. 1, p. 351–372, 2017. DOI: 10.1007/s00181-016-1158-5.

CRISTIANINI, Nello; SHAWE-TAYLOR, John. **An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods**. New York, NY, USA: Cambridge University Press, 2000. ISBN 0-521-78019-5.

CRYER, Jonathan D.; CHAN, Kung-Sik. **Time Series Analysis: With Applications in R**. 2. ed. New York: Springer, Nov. 2010. ISBN 978-0-387-75958-6.

D'ELIA, A.; PICCOLO, D. Maximum likelihood estimation of ARFIMA models with a Box-Cox transformation. **Statistical Methods and Applications**, v. 12, n. 3, p. 259–275, 2003.

DAVYDENKO, A.; FILDES, R. Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. **International Journal of Forecasting**, v. 29, n. 3, p. 510–522, 2013.

DIEBOLD, Francis X.; LOPEZ, Jose A. 8 Forecast evaluation and combination. In: STATISTICS, BT - Handbook of (Ed.). [S.l.]: Elsevier, 1996. v. 14. (Statistical Methods in Finance). p. 241–268. Visited on: 29 Oct. 2016.

DIEBOLD, Francis X.; MARIANO, Robert S. Comparing Predictive Accuracy. **Journal of Business & Economic Statistics**, v. 20, n. 1, p. 134–144, Jan. 2002. ISSN 0735-0015. DOI: 10.1198/073500102753410444. Visited on: 29 Oct. 2016.

DITTMANN, I.; GRANGER, C.W.J. Properties of nonlinear transformations of fractionally integrated processes. **Journal of Econometrics**, v. 110, n. 2, p. 113–133, 2002.

DOUC, R.; FOKIANOS, K.; MOULINES, E. Asymptotic properties of quasi-maximum likelihood estimators in observation-driven time series models. **Electronic Journal of Statistics**, v. 11, n. 2, p. 2707–2740, 2017. DOI: 10.1214/17-EJS1299.

DUDEK, G. Neural networks for pattern-based short-term load forecasting: A comparative study. **Neurocomputing**, v. 205, p. 64–74, 2016.

ESLING, Philippe; AGON, Carlos. Time-series data mining. **ACM Computing Surveys**, v. 45, n. 1, p. 1–34, Nov. 2012.

EUGSTER, Manuel J. A.; LEISCH, Friedrich. Bench Plot and Mixed Effects Models: First Steps toward a Comprehensive Benchmark Analysis Toolbox. In: _____. **Compstat 2008—Proceedings in Computational Statistics**. [S.l.]: Physica Verlag, Heidelberg, Germany, 2008. p. 299–306.

FOX, John. **Applied Regression Analysis and Generalized Linear Models**. 3rd. Los Angeles: SAGE Publications, Inc, Apr. 2015. ISBN 978-1-4522-0566-3.

FRYZLEWICZ, P.; NASON, G.P. Haar-Fisz estimation of evolutionary wavelet spectra. **Journal of the Royal Statistical Society. Series B: Statistical Methodology**, v. 68, n. 4, p. 611–634, 2006.

FRYZLEWICZ, P.; SAPATINAS, T.; RAO, S.S. A Haar-Fisz technique for locally stationary volatility estimation. **Biometrika**, v. 93, n. 3, p. 687–704, 2006.

GAO, J. a b et al. Traffic flow forecasting based on wavelet neural network optimized by GA. In: CHINESE Control Conference, CCC. [S.l.: s.n.], 2013. p. 8708–8712.

GIL-ALANA, L. A. Measuring the memory parameter on several transformations of asset returns. **International Journal of Theoretical and Applied Finance**, v. 8, n. 6, p. 675–691, 2005.

GIL-ALANA, L. A. Modelling the US real GNP with fractionally integrated techniques. **Applied Economics**, v. 36, n. 8, p. 873–879, 2004.

GIL-ALANA, L. Alberiko; JIANG, L. The purchasing power parity hypothesis in the us-china relationship: Fractional integration, time variation and data frequency. **International Journal of Finance and Economics**, v. 18, n. 1, p. 82–92, 2013.

GIRISH, G. P. a; TIWARI, A. K. b. A comparison of different univariate forecasting models for spot electricity price in India. **Economics Bulletin**, v. 36, n. 2, p. 1039–1057, 2016.

GOOS-BRASIL. **PIRATA Dataset**. [S.l.], 2015.

GOSPODINOV, N.; GAVALA, A.; JIANG, D. Forecasting volatility. **Journal of Forecasting**, v. 25, n. 6, p. 381–400, 2006. DOI: 10.1002/for.993.

GOURIEROUX, C.; JASIAK, J. **Nonlinear persistence and copersistence**. [S.l.: s.n.], 2010. DOI: 10.1057/9780230295216_4.

GUJARATI, Damodar. **Basic Econometrics**. 4. ed. Boston; Montreal: McGraw-Hill/Irwin, Mar. 2002. ISBN 978-0-07-247852-5.

HALDRUP, N. a; NIELSEN, M. A b. Estimation of fractional integration in the presence of data noise. **Computational Statistics and Data Analysis**, v. 51, n. 6, p. 3100–3114, 2007.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data Mining: Concepts and Techniques**. 3. ed. Haryana, India; Burlington, MA: Morgan Kaufmann, July 2011. ISBN 978-93-80931-91-3.

HANSSENS, Dominique M.; PARSONS, Leonard J.; SCHULTZ, Randall L. **Market Response Models: Econometric and Time Series Analysis**. 2nd edition. Boston, Mass.: Springer, Jan. 2003. ISBN 978-1-4020-7368-7.

HAYKIN, Simon. **Neural Networks: A Comprehensive Foundation**. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HAYKIN, Simon O. **Neural Networks and Learning Machines**. 3 edition. New York: Prentice Hall, Nov. 2008. ISBN 978-0-13-147139-9.

HENDRY, D. F. Robustifying forecasts from equilibrium-correction systems. **Journal of Econometrics**, v. 135, n. 1-2, p. 399–426, 2006.

HIPEL, K. W. Geophysical Model Discrimination Using the Akaike Information Criterion. **IEEE Transactions on Automatic Control**, v. 26, n. 2, p. 358–378, 1981.

HUANG, Guang-Bin; ZHU, Qin-Yu; SIEW, Chee-Kheong. Extreme learning machine: Theory and applications. **Neurocomputing**, v. 70, n. 1, p. 489–501, 2006. Neural Networks. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2005.12.126. Available from: ¡http://www.sciencedirect.com/science/article/pii/S0925231206000385¿.

HUNT, Tyler. **ModelMetrics: Rapid Calculation of Model Metrics**. [S.l.: s.n.], Nov. 2018. Available from: ¡https://CRAN.R-project.org/package=ModelMetrics¿. Visited on: 14 Jan. 2019.

HYNDMAN, Rob J.; ATHANASOPOULOS, George. **Forecasting: principles and practice**. S.l.: OTexts, Oct. 2013. ISBN 978-0-9875071-0-5.

HYNDMAN, Rob J.; KHANDAKAR, Yeasmin. Automatic Time Series Forecasting: The forecast Package for R. **Journal of Statistical Software**, v. 27, n. 3, p. 1–22, July 2008. ISSN 1548-7660.

HYNDMAN, Rob J et al. A state space framework for automatic forecasting using exponential smoothing methods. **International Journal of Forecasting**, v. 18, n. 3, p. 439–454, July 2002. ISSN 0169-2070. DOI: 10.1016/S0169-2070(01)00110-8. Visited on: 29 Oct. 2016.

IPEA. **Ipeadata. Macroeconomic and regional data**. [S.l.], 2017.

ISMAIL, M.T.; AWAJAN, A.M. A new hybrid approach EMD-EXP for short-term forecasting of daily stock market time series data. **Electronic Journal of Applied Statistical Analysis**, v. 10, n. 2, p. 307–327, 2017. DOI: 10.1285/i20705948v10n2p307.

JAMALMANESH, A. et al. Prediction of hydropower energy price using gómes-maravall seasonal model. **International Journal of Energy Economics and Policy**, v. 8, n. 2, p. 81–88, 2018.

JAMES, C.; MURTHY, H. A. Decoupling non-stationary and stationary components in long range network time series in the context of anomaly detection. In: PROCEEDINGS - Conference on Local Computer Networks, LCN. [S.l.: s.n.], 2012. p. 76–84.

JARA, E. C. Long memory time series forecasting by using genetic programming. **Genetic Programming and Evolvable Machines**, v. 12, n. 4, p. 429–456, 2011.

JOO, T.W.; KIM, S.B. Time series forecasting based on wavelet filtering. **Expert Systems with Applications**, v. 42, n. 8, p. 3868–3874, 2015.

KIM, Donghoh; OH, Hee-Seok. EMD: A Package for Empirical Mode Decomposition and Hilbert Spectrum. **The R Journal**, v. 1, n. 1, p. 40–46, 2009.

KO, K. a; VANNUCCI, M. b. Bayesian wavelet analysis of autoregressive fractionally integrated moving-average processes. **Journal of Statistical Planning and Inference**, v. 136, n. 10, p. 3415–3434, 2006.

KO, K.; VANNUCCI, M. Bayesian wavelet-based methods for the detection of multiple changes of the long memory parameter. **IEEE Transactions on Signal Processing**, v. 54, n. 11, p. 4461–4470, 2006.

KUMAR, Arun et al. Model Selection Management Systems: The Next Frontier of Advanced Analytics. **SIGMOD Rec.**, v. 44, n. 4, p. 17–22, May 2016. ISSN 0163-5808. DOI: 10.1145/2935694.2935698. Visited on: 8 Nov. 2016.

LAHMIRI, S. Interest rate next-day variation prediction based on hybrid feedforward neural network, particle swarm optimization, and multiresolution techniques. **Physica A: Statistical Mechanics and its Applications**, v. 444, p. 388–396, 2016.

LENDASSE, A. et al. Time series prediction competition: The CATS benchmark. **Neurocomputing**, v. 70, n. 13-15, p. 2325–2329, 2007.

LESSMANN, S. et al. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. **IEEE Transactions on Software Engineering**, v. 34, n. 4, p. 485–496, July 2008. ISSN 0098-5589. DOI: 10.1109/TSE.2008.35.

LI, R. et al. Hierarchical decomposition method and combination forecasting scheme for access load on public map service platforms. **Future Generation Computer Systems**, v. 87, p. 213–227, 2018. DOI: 10.1016/j.future.2018.03.031.

LIAW, Andy; WIENER, Matthew. Classification and Regression by randomForest. **R News**, v. 2, n. 3, p. 18–22, 2002. Available from: ¡https://CRAN.R-project.org/doc/Rnews/¿.

LIU, S.; GU, S.; PENG, J. Self-adaptive processing and forecasting algorithm for univariate linear time series. **Chinese Journal of Electronics**, v. 26, n. 6, p. 1147–1153, 2017. DOI: 10.1049/cje.2017.09.027.

LJUNG, G.M.; LEDOLTER, J.; ABRAHAM, B. George Box's contributions to time series analysis and forecasting. **Applied Stochastic Models in Business and Industry**, v. 30, n. 1, p. 25–35, 2014.

LOS, Cornelis. **Financial Market Risk: Measurement and Analysis**. 1. ed. London; New York: Routledge, Aug. 2006. ISBN 978-0-415-77113-9.

LUUKKO, Perttu J.J.; HELSKE, Jouni; RÄSÄNEN, Esa. Introducing libeemd: A program package for performing the ensemble empirical mode decomposition. **Computational Statistics**, v. 31, n. 2, p. 545–557, June 2016. ISSN 1613-9658. DOI: 10.1007/s00180-015-0603-9. Available from: ¡https://doi.org/10.1007/s00180-015-0603-9¿.

MACHADO, E. et al. Exploring machine learning methods for the Star/Galaxy Separation Problem. In: PROCEEDINGS of the International Joint Conference on Neural Networks. [S.l.: s.n.], 2016. 2016-October, p. 123–130. DOI: 10.1109/IJCNN.2016.7727189. Available from: ¡https://www.scopus.com/inward/record.uri?eid=2-s2.0-85007238108&doi=10.1109%2fIJCNN.2016.7727189&partnerID=40&md5=980ebb6e4d037c098e9f3a9ad9228025¿.

MAIER, H. R.; DANDY, G. C. Neural network based modelling of environmental variables: A systematic approach. **Mathematical and Computer Modelling**, v. 33, 6–7, p. 669–682, Mar. 2001. ISSN 0895-7177.

MARROCU, E. An investigation of the effects of data transformation on nonlinearity. **Empirical Economics**, v. 31, n. 4, p. 801–820, 2006.

MAYNARD, A.; SMALLWOOD, A.; WOHAR, M.E. Long Memory Regressors and Predictive Testing: A Two-stage Rebalancing Approach. **Econometric Reviews**, v. 32, n. 3, p. 318–360, 2013.

MEYER, David et al. **e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien**. [S.l.: s.n.], July 2018. Available from: ¡https://CRAN.R-project.org/package=e1071¿. Visited on: 14 Jan. 2019.

MILIONIS, A.E. The importance of variance stationarity in economic time series modelling. A practical approach. **Applied Financial Economics**, v. 14, n. 4, p. 265–278, 2004. DOI: 10.1080/0960310042000220040.

MILLS, T.C.; MARKELLOS, R.N. **The econometric modelling of financial time series**. [S.l.: s.n.], 2008.

MINU, K.K.; LINEESH, M.C.; JESSY JOHN, C. Wavelet neural networks for nonlinear time series analysis. **Applied Mathematical Sciences**, v. 4, n. 49-52, p. 2485–2495, 2010.

MOGHRAM, I.; RAHMAN, S. Analysis and evaluation of five short-term load forecasting techniques. **IEEE Transactions on Power Systems**, v. 4, n. 4, p. 1484–1491, 1989.

MORANA, C. Multivariate modelling of long memory processes with common components. **Computational Statistics and Data Analysis**, v. 52, n. 2, p. 919–934, 2007.

MOREIRA, Leonardo et al. On Evaluating Data Preprocessing Methods for Machine Learning Models for Flight Delays. In: 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018. [S.l.: s.n.], 2018. p. 1–8. DOI: 10.1109/IJCNN.2018.8489294. Available from: ¡https://doi.org/10.1109/IJCNN.2018.8489294¿.

MORENO, Alberto Vico; RIVAS, Antonio Jesus Rivera; GODOY, Maria Dolores Perez. **predtoolsTS: Time Series Prediction Tools**. [S.l.: s.n.], Apr. 2018. Available from: ¡https://CRAN.R-project.org/package=predtoolsTS¿. Visited on: 22 Jan. 2019.

MOUSELIMIS, Lampros; GOSSO, Alberto. **elmNNRcpp: The Extreme Learning Machine Algorithm**. [S.l.: s.n.], July 2018. Available from: ¡https://CRAN.R-project.org/package=elmNNRcpp¿. Visited on: 14 Jan. 2019.

MURPHY, Kevin P. **Machine Learning: A Probabilistic Perspective**. 1. ed. Cambridge, MA: The MIT Press, Aug. 2012. ISBN 978-0-262-01802-9.

NACHANE, D.; CLAVEL, J.G. Forecasting interest rates: A comparative assessment of some second-generation nonlinear models. **Journal of Applied Statistics**, v. 35, n. 5, p. 493–514, 2008.

NGENE, G.; MUNGAI, A.N.; LYNCH, A.K. Long-Term Dependency Structure and Structural Breaks: Evidence from the U.S. Sector Returns and Volatility. **Review of Pacific Basin Financial Markets and Policies**, v. 21, n. 2, 2018. DOI: 10.1142/S021909151850008X.

NN3. **The NN3 Competition**. [S.l.], 2007.

NN5. **The NN5 Competition**. [S.l.], 2008.

NURY, Ahmad Hasan; HASAN, Khairul; ALAM, Md Jahir Bin. Comparative study of wavelet-ARIMA and wavelet-ANN models for temperature time series data in northeastern Bangladesh. **Journal of King Saud University - Science**, v. 29, n. 1, p. 47–61, Jan. 2017. ISSN 1018-3647.

OGASAWARA, E. et al. Adaptive Normalization: A novel data normalization approach for non-stationary time series. In: THE 2010 International Joint Conference on Neural Networks (IJCNN). [S.l.: s.n.], July 2010. p. 1–8.

OGASAWARA, E. et al. An algebraic approach for data-centric scientific workflows. **Proceedings of the VLDB Endowment**, v. 4, p. 1328–1339, 2011. Available from: ¡https://www.scopus.com/inward/record.uri?eid=2-s2.0-84856343648&partnerID=40&md5=9ab2db75c2a67bccaa5005a4fb5b32e2¿.

OGASAWARA, E. et al. Neural networks cartridges for data mining on time series. In: INTERNATIONAL Joint Conference on Neural Networks, 2009. IJCNN 2009. [S.l.: s.n.], June 2009. p. 2302–2309. DOI: 10.1109/IJCNN.2009.5178615.

OMTZIGT, P.; PARUOLO, P. Impact factors. **Journal of Econometrics**, v. 128, n. 1, p. 31–68, 2005.

OTOK, B.W.; SUHARTONO. Development of rainfall forecasting model in Indonesia by using ASTAR, transfer function, and ARIMA methods. **European Journal of Scientific Research**, v. 38, n. 3, p. 386–395, 2009.

PAI, J.S.; RAVISHANKER, N. Fast bayesian estimation for VARFIMA processes with stable errors. **Journal of Statistical Theory and Practice**, v. 4, n. 4, p. 663–677, 2010. DOI: 10.1080/15598608.2010.10412011.

PALMA, W. **Long-Memory Time Series: Theory and Methods**. [S.l.: s.n.], 2006.

PERCIVAL, D.B.; MONDAL, D. A Wavelet Variance Primer. **Handbook of Statistics**, v. 30, p. 623–657, 2012.

POGGIO, Tomaso; GIROSI, Federico. **A Theory of Networks for Approximation and Learning**. Cambridge, MA, USA, 1989.

PYLE, Dorian. **Data Preparation for Data Mining**. 1. ed. San Francisco, Calif: Morgan Kaufmann, Apr. 1999. ISBN 978-1-55860-529-9.

R DEVELOPMENT CORE TEAM. **R: A Language and Environment for Statistical Computing**. Vienna, Austria: R Foundation for Statistical Computing, 2008. ISBN 3-900051-07-0.

RAMEY, John A. **sortinghat: sortinghat**. [S.l.: s.n.], Dec. 2013. Available from: ¡https://cran.r-project.org/web/packages/sortinghat/index.html¿. Visited on: 6 Feb. 2017.

RISSANEN, Jorma. Strong optimality of the normalized ML models as universal codes and information in data. **IEEE Transactions on Information Theory**, v. 47, n. 5, p. 1712–1717, 2001.

RODRIGUEZ, G. Selecting between autoregressive conditional heteroskedasticity models: An empirical application to the volatility of stock returns in Peru. **Revista de Analisis Economico**, v. 32, n. 1, p. 69–94, 2017. DOI: 10.4067/S0718-88702017000100069.

ROSHAN, W.D.S.; GOPURA, R.A.R.C.; JAYASEKARA, A.G.B.P. Financial forecasting based on artificial neural networks: Promising directions for modeling. In: 2011 6th International Conference on Industrial and Information Systems, ICIIS 2011 - Conference Proceedings. [S.l.: s.n.], 2011. p. 322–327.

SADAEI, H.J. et al. Combining ARFIMA models and fuzzy time series for the forecast of long memory time series. **Neurocomputing**, v. 175, p. 782–796, 2016.

SALLES, Rebecca Pontes; OGASAWARA, Eduardo. **TSPred: Functions for Benchmarking Time Series Prediction Prediction**. [S.l.], 2018.

SALLES, Rebecca et al. A framework for benchmarking machine learning methods using linear models for univariate time series prediction. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017. [S.l.: s.n.], 2017. p. 2338–2345. DOI: 10.1109/IJCNN.2017.7966139. Available from: ¡https://doi.org/10.1109/IJCNN.2017.7966139¿.

SALLES, Rebecca et al. Evaluating Linear Models as a Baseline for Time Series Imputation. In: XXX Brazilian Symposium on Databases. Petrópolis, RJ: [s.n.], Oct. 2015.

SALLES, Rebecca et al. Evaluating temporal aggregation for predicting the sea surface temperature of the Atlantic Ocean. **Ecological Informatics**, v. 36, p. 94–105, Nov. 2016. ISSN 1574-9541. DOI: 10.1016/j.ecoinf.2016.10.004. Available from: ¡http://www.sciencedirect.com/science/article/pii/S1574954116301819¿. Visited on: 20 Jan. 2019.

SALLES, Rebecca et al. Nonstationary time series transformation methods: An experimental review. **Knowledge-Based Systems**, v. 164, p. 274–291, Jan. 2019. ISSN 0950-7051. DOI: 10.1016/j.knosys.2018.10.041. Available from: ¡http://www.sciencedirect.com/science/article/pii/S0950705118305343¿. Visited on: 20 Jan. 2019.

SARMA, U.K.; SINHA, S.; BASU, T.K. Medium range forecasting of power system load (energy) demand. **International Journal of Systems Science**, v. 18, n. 9, p. 1691–1702, 1987.

SHU, M.-H. et al. Forecasting cargo throughput with modified seasonal ARIMA models. **WSEAS Transactions on Mathematics**, v. 13, p. 171–181, 2014.

SHUMWAY, Robert H.; STOFFER, David S. **Time Series Analysis and Its Applications: With R Examples**. 4. ed. New York, NY: Springer, Apr. 2017. ISBN 978-3-319-52451-1.

SONG, W. et al. Geographic spatiotemporal big data correlation analysis via the Hilbert–Huang transformation. **Journal of Computer and System Sciences**, v. 89, p. 130–141, 2017. DOI: 10.1016/j.jcss.2017.05.010.

STEFANAKOS, C.; SCHINAS, O. Forecasting bunker prices; a nonstationary, multivariate methodology. **Transportation Research Part C: Emerging Technologies**, v. 38, p. 177–194, 2014.

STENSHOLT, E.; TJØSTHEIM, D. Factorizing multivariate time series operators. **Journal of Multivariate Analysis**, v. 11, n. 2, p. 244–249, 1981.

SUN, G. et al. A carbon price forecasting model based on variational mode decomposition and spiking neural networks. **Energies**, v. 9, n. 1, 2016.

TAYLOR, Karl E. Summarizing multiple aspects of model performance in a single diagram. **Journal of Geophysical Research: Atmospheres**, v. 106, n. D7, p. 7183–7192, 2001.

TSAY, Ruey S. **Analysis of Financial Time Series**. 3. ed. Cambridge, Mass: Wiley, Aug. 2010. ISBN 978-0-470-41435-4.

VENABLES, W. N.; RIPLEY, B. D. **Modern Applied Statistics with S**. Fourth. New York: Springer, 2002. ISBN 0-387-95457-0. Available from: ¡http://www.stats.ox.ac.uk/pub/MASS4¿.

VENKATARAMAN, Shivaram et al. SparkR: Scaling R Programs with Spark. In: PROCEEDINGS of the 2016 International Conference on Management of Data. San Francisco, California, USA: ACM, 2016. (SIGMOD '16), p. 1099–1104. ISBN 978-1-4503-3531-7. DOI: 10.1145/2882903.2903740. Available from: ¡http://doi.acm.org/10.1145/2882903.2903740¿.

WANG, H. et al. A novel work zone short-term vehicle-type specific traffic speed prediction model through the hybrid EMD–ARIMA framework. **Transportmetrica B**, v. 4, n. 3, p. 159–186, 2016.

WICKHAM, Hadley. **Advanced R**. 1 edition. Boca Raton, FL: Routledge, Sept. 2014. ISBN 978-1-4665-8696-3.

WOLPERT, D.H.; MACREADY, W.G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, Apr. 1997. ISSN 1089-778X. DOI: 10.1109/4235.585893.

WU, Z. et al. On the trend, detrending, and variability of nonlinear and nonstationary time series. **Proceedings of the National Academy of Sciences of the United States of America**, v. 104, n. 38, p. 14889–14894, 2007.

XI, L. et al. Prediction model of non-stationary time series parameters for a complex blending process. In: 26TH Chinese Control and Decision Conference, CCDC 2014. [S.l.: s.n.], 2014. p. 1027–1030.

XIE, C.; BIJRAL, A.; FERRES, J.L. NonSTOP: A NonSTationary Online Prediction Method for Time Series. **IEEE Signal Processing Letters**, 2018. DOI: 10.1109/LSP.2018.2867724.

XIONG, H.; SHANG, P.; BIAN, S. Detecting intrinsic dynamics of traffic flow with recurrence analysis and empirical mode decomposition. **Physica A: Statistical Mechanics and its Applications**, v. 474, p. 70–84, 2017. DOI: 10.1016/j.physa.2017.01.060.

YANG, W. a; ZURBENKO, I. b. Nonstationarity. **Wiley Interdisciplinary Reviews: Computational Statistics**, v. 2, n. 1, p. 107–115, 2010.

YEO, I.N.-K.; JOHNSON, R.A. A new family of power transformations to improve normality or symmetry. **Biometrika**, v. 87, n. 4, p. 954–959, 2000.

ZAHARIA, Matei et al. Apache Spark: A Unified Engine for Big Data Processing. **Commun. ACM**, ACM, New York, NY, USA, v. 59, n. 11, p. 56–65, Oct. 2016. ISSN 0001-0782. DOI: 10.1145/2934664. Available from: ¡http://doi.acm.org/10.1145/2934664¿.

ZHANG, H. et al. A multivariate short-term traffic flow forecasting method based on wavelet analysis and seasonal time series. **Applied Intelligence**, v. 48, n. 10, p. 3827–3838, 2018. DOI: 10.1007/s10489-018-1181-7.

ZHANG, N.; LIN, A.; SHANG, P. Multidimensional k-nearest neighbor model based on EEMD for financial time series forecasting. **Physica A: Statistical Mechanics and its Applications**, v. 477, p. 161–173, 2017. DOI: 10.1016/j.physa.2017.02.072.