TSPred: A framework for nonstationary time series prediction

Rebecca Salles
CEFET/RJ
rebeccapsalles@acm.org

Esther Pacitti
INRIA & University of Montipellier
pacitti@lirmm.fr

Eduardo Bezerra
CEFET/RJ
ebezerra@cefet-rj.br

Fabio Porto
LNCC
fporto@lncc.br

Eduardo Ogasawara
CEFET/RJ
eogasawara@ieee.org

Abstract

The nonstationary time series prediction is challenging since it demands knowledge of both data transformation and prediction methods. This paper presents TSPred, a framework for nonstationary time series prediction. It differs from the mainstream frameworks since it establishes a prediction process that seamlessly integrates nonstationary time series transformations with state-of-the-art statistical and machine learning methods. It is made available as an R-package, which provides functions for defining and conducting time series prediction, including data pre(post)processing, decomposition, modeling, prediction, and accuracy assessment. Besides, TSPred enables user-defined methods, which significantly expands the applicability of the framework.

***K*eywords** time series · prediction · nonstationarity · machine learning · preprocessing · transform

1 Introduction

The prediction of time series has gained more attention among researchers. Many time series prediction methods have been developed and can be found in the literature [2]. Several state-of-the-art methods are based on machine learning, especially for long and potentially high-frequent time series. Unfortunately, most of these methods tend to be optimistic regarding their assumptions over the time series.

A common assumption is the property of stationarity [2]. In a stationary time series, statistical properties (mean, variance and covariance) do not change over time, which is not common for most real datasets. Generally, nonstationarity demands transformation methods to coerce the time series into stationarity [18]. Such methods can have different features depending on the implementation of the data properties they consider. Choosing an adequate method for a particular time series application is not a simple task.

Additionally, nonstationarity can lead to exploring many data transformation and prediction methods for building prediction models. Moreover, the choice of an appropriate transformation setup for a particular time series is not straightforward. The benchmarking of different candidate combinations helps select an appropriate setup for predicting a nonstationary time series.

In this context, this paper presents *TSPred*, a framework for nonstationary time series prediction. It specializes in integrating data transformation methods and machine learning methods (MLM) to predict long time series with potentially high frequency. *TSPred* is made available as an R-package [19]. It is the first tool to seamlessly integrate a broad range of transformation methods [18] and state-of-the-art statistical and machine learning prediction methods for addressing nonstationary time series. The package automates the time series prediction process and parameterization while also enabling user-defined prediction methods and data transformations. The features provided by *TSPred* are shown to be competitive regarding time series prediction accuracy.

Besides this introduction, Section 2 gives background on the time series prediction problem. Section 3 presents *TSPred*, while Sections 4 and 5 give empirical results and illustrate its usage. Finally, Section 6 concludes.

2 Problem and Background

A general nonstationary time series prediction process usually encompasses five steps [17]. The first refers to acquiring the training and evaluation datasets and applying nonstationary time series transformation methods. The second and third refer to training the time series prediction model and applying it to an evaluation set to obtain predictions. Forth corresponds to reversing transformations applied in the first step. Finally, the fifth is the evaluation of prediction errors and model fitness.

This process provides a systematic way of predicting a time series based on a particular setup of preprocessing and prediction methods. It also focuses on prediction/model evaluation, that is, evaluating the accuracy of prediction and the fitness of a model. Such evaluation may indicate a demand for refining and perfecting the preprocessing-modeling setup and its parameters to obtain a more accurate model. This process may be repeated if the evaluated time series prediction model does not reach the desired accuracy. This process enables benchmarking different preprocessing-modeling setups.

Several authors focused on the task of benchmarking different models in many different domains. Ramey [16] and Lessmann et al. [12] developed frameworks for benchmarking classification models and algorithms. Moreover, Bischl et al. [1] and Eugster and Leisch [6] developed the R-packages *mlr* and *benchmark*, respectively, which provide tools for executing automated experiments when benchmarking a set of models for data mining tasks such as classification and regression. These packages are designed to support tabular data and focus on benchmarking based on plot visualization.

Hyndman and Khandakar [7] and Hyndman et al. [8] present frameworks for automatic forecasting using mainly statistical models such as autoregressive integrated moving average (ARIMA) and exponential smoothing state space model (ETS). Hyndman and Khandakar [7] 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 [14] also facilitates time series prediction using simple differencing (DIF) and Box-Cox transform (BCT). Furthermore, we observed three worth mentioning works. Diebold and Mariano [5] propose various tests to compare the predictive accuracy of two different prediction models. Diebold and Lopez [4] propose an ensemble approach using different prediction models. Kumar et al. [10] propose a class of analytics systems to manage model selection using key ideas from data management research.

All in all, several works present frameworks and tools for MLM performance assessment. Nonetheless, to the best of our knowledge, no work proposes and implements a framework for seamless integration of transformation and time series prediction methods focusing on addressing nonstationary properties. Such framework contributes to the benchmarking of preprocessing-modeling setups for a particular nonstationary time series prediction application.

3 Framework and Functionalities

The main functionality modules representing the concept and structure of the framework are depicted in Figure 1. Together, they represent a particular time series prediction application and encapsulate the process described in Section 2 and all the elements and steps necessary to fulfill it. *TSPred* has three main functionality modules: *Preprocessing & Postprocessing*, *Modeling & Predicting*, and *Evaluating*.

The first module is responsible for preprocessing/transforming and postprocessing/reverse transforming time series. It includes the implementation of the main nonstationary time series transformation methods [18], being either mapping-based, namely the logarithmic transform (LT), BCT, percentage change transform (PCT), moving average smoother (MAS), and DIF, or splitting-based, such as empirical mode decomposition (EMD) and wavelet transform (WT). Furthermore, it implements other relevant preprocessing methods for time series prediction with MLM: partitioning time series data into training and evaluation datasets, subsetting the time series data into sliding windows, handling of missing values, and data normalization via Min-max normalization (MM) and Adaptive normalization (AN) methods [15].

The *Modeling & Predicting* module is responsible for modeling and predicting a time series based on a particular time series model. These tasks are specialized for either statistical or machine learning models. For the latter, the framework is prepared to perform any necessary machine learning life-cycle tasks during the training and prediction steps, including coercing data into sliding windows and performing normalization and transformation of the input data. This module includes the implementation of the statistical models: ARIMA, Holt-Winter's exponential smoothing (HW), theta forecasting (TF), and ETS. *MLM* models include feed-forward neural network (NNET), random forest regression (RFrst), radial basis function network (RBF), support vector machine (SVM), multilayer perceptron network (MLP), and extreme learning machines network (ELM). Furthermore, the module provides deep learning

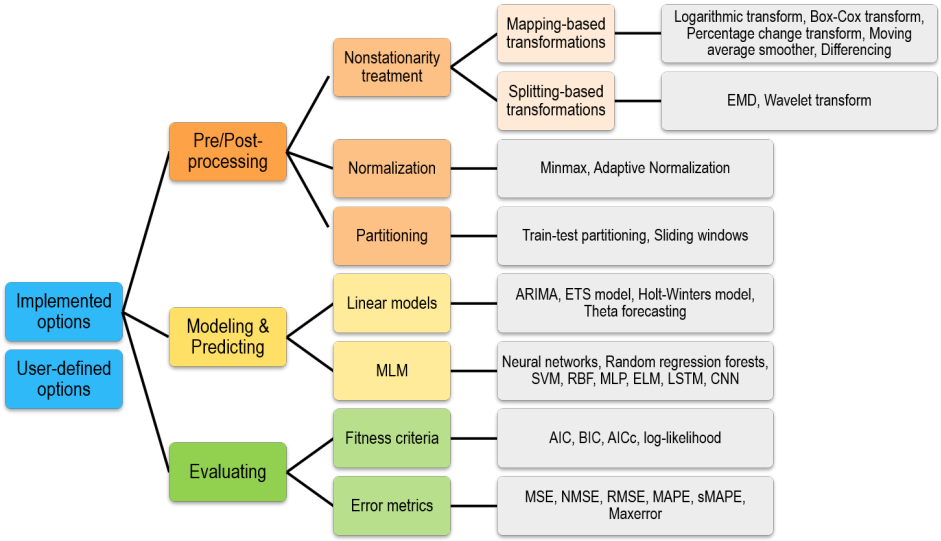

Figure 1: TSPred functionality modules and pre-implemented algorithms

models available in the *TensorFlow* library, namely convolutional neural network (CNN) and long short-term memory neural network (LSTM).

Finally, the *Evaluating* module is responsible for assessing the model fitness and quality of predictions for the given time series based on a particular metric. These tasks are specialized for computing either prediction accuracy (error) measures or model fitting criteria. Among the available prediction accuracy measures, it includes Mean Square Error (MSE), symmetric MAPE (sMAPE), and maximal error. It also includes model fitness criteria such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and log-likelihood [3].

TSPred can integrate the described modules in a *workflow*, encompassing the five steps described in Section 2. Ultimately, the package provides the means to perform the *benchmarking* of several prediction models. It is important to remark that although providing several pre-implemented options, *TSPred* design enables the user to define and apply customized time series prediction methods.

Moreover, the package provides several automatized features that can be useful for any time series prediction application. Among them, some of the main features are: (i) seamless recursive combination of two or more transformation methods; (ii) seamless integration of splitting-based transformation methods to the prediction process [18], which demands the combination of predictions for each component resulting from data decomposition (first package to include this approach); (iii) transformation/model parameter selection; (iv) multistep-ahead or one-step-ahead predictions; (v) rolling origin evaluation [9] for both statistical and machine learning models; and (vi) machine-learning life-cycle tasks performed during training and prediction steps. It means that data normalization and transformation of sliding windows are seamlessly conducted during machine learning model training.

The framework is implemented in R using the S3 class system [20]. *TSPred* is currently in version 5.1 and is available at The Comprehensive R Archive Network (CRAN). For more implementation details, please refer to the *TSPred* documentation [19].

4 Empirical Results

The *TSPred* package includes datasets provided by time series prediction competitions to explore different time series prediction applications. The CATS Competition dataset [11] is one of them. It presents an artificial time series with 5000 observations, among which 100 are missing. The missing observations are grouped into five non-consecutive gaps of 20. Each subset of known data that precedes a gap may be considered a different time series to be modeled. Although prediction techniques can be used in each gap, interpolation techniques might influence comparisons in the first four gaps. In contrast, the fifth gap can only be solved using a prediction process. Table 1 presents the top-3 ranking of MSE prediction errors produced by the competitors of CATS for the fifth gap of missing observations.

The same prediction experiment was performed using the *TSPred* package and its implemented resources. Table 2 presents the adopted approaches combining data preprocessing and data modeling, either by the machine learning

Table 1: CATS top 3 methods for the fifth gap of observations [11]

MSE	Model
597	Recurrent Neural Networks
656	Kalman Smoother
672	BYY Harmony Learning Based Mixture of Experts Model

Table 2: MSE for the fifth gap of observations using TSPred

MSE	Nonstationary time series transform	Model
130	Empirical Mode Decomposition	Support Vector Machines
278	None	ARIMA (baseline)
325	None	Convolutional Neural Networks
358	None	Support Vector Machines
461	Empirical Mode Decomposition	Convolutional Neural Networks

model SVM or CNN. It follows that the time series in CATS are mostly nonstationary, which knowingly poses a challenge to prediction tasks [18]. For that reason, we adopted a data preprocessing step based on applying splitting-based nonstationary time series transform EMD. Furthermore, we adopted the statistical ARIMA model as a baseline [17]. The results from each approach are ranked based on MSE prediction errors.

TSPred is competitive when comparing them to the errors produced by state-of-the-art prediction models adopted by the top-ranked CATS competitors. Furthermore, the demand for adopting a suitable baseline model is noticeable given that the statistical ARIMA model produced smaller errors than the presented in Table 1. In particular, the deep learning model CNN could not outperform the baseline model. The same is valid for the SVM model by itself. However, introducing a preprocessing step designed to address the nonstationarity in the CATS time series resulted in the smaller prediction errors among the 22 presented approaches.

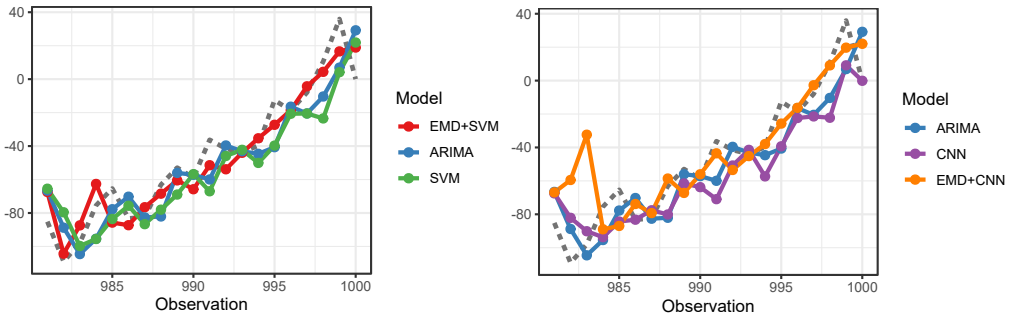

Figure 2: Visualization of TSPred predictions for block 5

The predictions from Table 2 are depicted in Figure 2. While the predictions from SVM and CNN seem closer to the baseline (ARIMA), their combination with EMD allowed a better representation of the actual observations. Nonetheless, CNN did not outperform SVM and yielded higher errors, especially for the first predictions.

TSPred enabled the application of all the different prediction approaches presented in Table 2. The results were obtained with few lines of code. The only parameter explicitly given was the length of sliding windows. All other parameters required by the models and the data transform were duly estimated by *TSPred*, resulting in more optimized prediction results. The package was designed to enable a seamless integration of prediction models, including state-of-the-art MLM and nonstationary time series preprocessing methods. The combination of preprocessing methods is also supported and may lead to even smaller prediction errors. For example, the BCT and EMD data transforms with SVM modeling led to a smaller MSE (120).

Besides the CATS time series, we present results obtained by *TSPred* over a time series dataset drawn from the M5 Accuracy competition held in 2020 [13]. The dataset contains real-world daily observations of retail sales (1,969 days or approximately 5.4 years, from 2011 to 2016). It was asked to the competitors that they predict the next 28

daily sales. The provided time series dataset can be grouped based on either location (store and state) or product-related information (department and category). The competition allowed the participants to use cross-sectional levels of aggregation for prediction evaluation. Henceforth we discuss prediction results for the time series of unit sales aggregated for each of the ten stores available (aggregation level 3) [13].

The prediction of the selected M5 time series dataset was performed with *TSPred*, using both *MLM* models (NNET, ELM, SVM, CNN, LSTM) and statistical models (ETS, ARIMA).We adopted a rolling origin scheme and one-step-ahead predictions for all adopted prediction models for a more robust evaluation. Moreover, different nonstationary time series transformation methods were used and combined to evaluate their impact on prediction. In that case, different transformation combination setups were adopted, including mapping-based methods (BCT, MAS, and DIF) and splitting-based methods (EMD and WT) [18]. *TSPred* automatically selected the parameters for all transformation methods. Data normalization was performed with either MM or AN, and sliding window length was set to 30 (based on the central limit theorem). All other parameters were set to default values. Besides, for the sake of discussion, transformation methods were not applied to CNN and LSTM. According to the M5 competition guidelines, the best time series prediction setups were ranked based on the Weighted Root Mean Squared Scaled Error (WRMSSE) metric. Table 3 presents some of the best evaluated results obtained by *TSPred* on the adopted M5 time series dataset. The simulated position of the results on the top-50 ranking of the M5 competition (regarding agg. level 3) is also given.

Table 3: WRMSSE obtained by TSPred for the M5 time series (agg. level 3)

WRMSSE	Nonstationary time series transform	Model	Rank
0.455	Box-Cox	Feed-forward neural network (NNET)	35
0.472	Box-Cox	Support Vector Machines (SVM)	46
0.491	Box-Cox	Extreme learning machines network (ELM)	48
0.590	None	Convolutional Neural Networks (CNN)	-
0.596	None	Long short-term memory neural network (LSTM)	-
0.635	None	ARIMA (baseline)	-
0.909	None	Exponential smoothing model (ETS) (baseline)	-

Overall, better results were given by the use of MM normalization and mapping-based transform BCT. As observed from Table 3, the gap between prediction accuracy performances of statistical models and *MLM* models increased in comparison to the results from CATS (Table 2). It is expected since *MLM* models are more competitive in scenarios of long and high-frequency time series, such as the M5 time series dataset. Statistical models did not make it to the top-50 ranking of the M5 competition. Nonetheless, the use of deep learning *MLM* models (LSTM, CNN) without transformation methods could not climb to the top-50 ranking in M5, despite being computationally expensive. On the other hand, NNET, SVM, and ELM (faster and less complex) models combined with nonstationary time series transformation methods yielded better prediction results.

As presented in Table 3, even using default setups, *TSPred* prediction results were competitive, featuring in the top-50 ranked prediction results. In that case, hyperparameter optimization can probably result in even better performances and higher rank positions. *TSPred* furthers such optimization via the benchmarking of several different prediction setups. Moreover, *TSPred* can incorporate the best approaches by defining user-customized methods and benefiting from the broad range of transformation methods.

5 Illustrative Examples

This section gives examples of *TSPred* usage. The first example corresponds to a time series prediction using the NNET model combined with different data transformation methods. In Listing 1 the *TSPred* R-package and the CATS dataset are loaded into the R programming environment. The components for the time series process can be defined separately for enabling reuse. An example is presented in Listing 1. First, data *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 sliding windows technique (SW) (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 NNET modeling and prediction with five input units in a single hidden layer. `proc_sw` and `proc_mm` are given as parameters. At last an *evaluating* object respective to MSE is obtained.

Listing 1: R example of the definition of components/steps of a time series prediction process in *TSPred*

```
#installing and loading TSPred package
> install.packages("TSPred")
> library("TSPred")

#loading CATS dataset
> data("CATS")

#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 2: 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[5]) %>%
                    preprocess(prep_test = TRUE) %>%
                    train() %>%
                    predict(input_test_data = TRUE) %>%
                    postprocess() %>%
                    evaluate()
```

The previously defined objects are used to define the time series prediction process by MLM as shown in Listing 2. The object `tspred_mlm` receives an instance of the *tspred* class. An instance of this class represents a particular time series prediction application. It encapsulates the process described in Section 2, including all its elements (attributes) and steps (class methods). 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 NNET modeling process with set parameters; and (iii) a list of evaluation metrics to be computed for prediction accuracy (MSE). The prediction process represented by `tspred_mlm` is applied for the fifth sequence of known values of the CATS series (`CATS[5]`) and executed step-by-step as in Listing 2 (or by the function `workflow`). It returns a *tspred* class object similar to `tspred_mlm` updated with output elements and selected parameters `tspred_mlm_res`. Different *tspred* objects containing different prediction setups can be benchmarked and ranked based on prediction evaluation criteria using the function `benchmark`. The implementation of user-defined methods is done by extending the subclasses of *TSPred*. For example, the user specifies the functions for training and prediction of the model they wish to implement and their respective parameters to define a new MLM model.

6 Conclusions

This paper presented *TSPred*. It focuses on automatizing the entire time series prediction process. It is made available as an R Package [19]. It seamlessly provides several automated features such as the combination of time series transformation methods in pipelines, prediction with decomposed time series, transformation and model parameter selection, multistep/one-step-ahead prediction, rolling origin evaluation, and the management of sliding windows. Several benchmark datasets from time series prediction competitions come bundled with *TSPred*, enabling users to practice data transformation and prediction methods, gaining confidence in the developed prediction models. Besides, the framework was designed to enable the user to implement their customized methods. Future updates will expand the range of implemented preprocessing methods, MLM, and evaluation metrics. The support for cross-learning and multivariate data is also envisaged.

Acknowledgements

The authors thank CNPq, CAPES (finance code 001), and FAPERJ for partially sponsoring this research.

References

[1] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. Mlr: Machine learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016. ISSN 1533-7928.

[2] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. *IIE Transactions (Institute of Industrial Engineers)*, 47(10):1053–1071, 2015.

[3] A. Davydenko and R. Fildes. Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. *International Journal of Forecasting*, 29(3):510–522, 2013.

[4] F. Diebold and J. Lopez. 8 Forecast evaluation and combination. *Handbook of Statistics*, 14:241–268, 1996.

[5] F. Diebold and R. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 20(1): 134–144, 2002.

[6] M. J. A. Eugster and F. Leisch. Bench Plot and Mixed Effects Models: First Steps toward a Comprehensive Benchmark Analysis Toolbox. In P. Brito, editor, *Compstat 2008*, pages 299–306. Physica Verlag, Heidelberg, Germany, 2008.

[7] R. Hyndman and Y. Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22, 2008.

[8] R. Hyndman, A. Koehler, R. Snyder, and S. Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18(3):439–454, 2002.

[9] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[10] A. Kumar, R. McCann, J. Naughton, and J. M. Patel. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *ACM SIGMOD Record*, 44(4):17–22, May 2016. ISSN 0163-5808.

[11] A. Lendasse, E. Oja, O. Simula, and M. Verleysen. Time series prediction competition: The CATS benchmark. *Neurocomputing*, 70(13-15):2325–2329, 2007.

[12] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4): 485–496, 2008.

[13] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*, 2020.

[14] A. V. Moreno, A. J. R. Rivas, and M. D. P. Godoy. predtoolsTS: Time Series Prediction Tools. Technical report, https://cran.r-project.org/package=predtoolsTS, 2018.

[15] E. Ogasawara, L. Martinez, D. De Oliveira, G. Zimbrão, G. Pappa, and M. Mattoso. Adaptive Normalization: A novel data normalization approach for non-stationary time series. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2010.

[16] J. A. Ramey. sortinghat: sortinghat. Technical report, https://cran.r-project.org/web/packages/sortinghat/index.html, 2013.

[17] R. Salles, L. Assis, G. Guedes, E. Bezerra, F. Porto, and E. Ogasawara. A framework for benchmarking machine learning methods using linear models for univariate time series prediction. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2017-May, pages 2338–2345, 2017.

[18] R. Salles, K. Belloze, F. Porto, P. Gonzalez, and E. Ogasawara. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291, 2019.

[19] R. P. Salles and E. Ogasawara. TSPred: Functions for Benchmarking Time Series Prediction. Technical report, https://cran.r-project.org/package=TSPred, 2021.

[20] H. Wickham. *Advanced R*. CRC Press, second edition, May 2019. ISBN 978-1-351-20130-8.