Progress in Artificial Intelligence (2019) 8:505–510
https://doi.org/10.1007/s13748-019-00193-z

REGULAR PAPER

predtoolsTS: R package for streamlining time series forecasting

Francisco Charte[1] · Alberto Vico[1] · María D. Pérez-Godoy[1] · Antonio J. Rivera[1]

Received: 29 March 2019 / Accepted: 17 May 2019 / Published online: 5 June 2019

Abstract
Time series forecasting is a field of interest in many areas. Classically, statistical methods have been used to address this problem. In recent years, machine learning (ML) algorithms have been also applied with satisfactory results. However, ML software packages are not skilled to deal with raw sequences of temporal data, and therefore, it is necessary to transform these time series. This paper presents **predtoolsTS**, an R package that provides a uniform interface for applying both statistical and ML methods to time series forecasting. **predtoolsTS** comprises four modules: preprocessing, modeling, prediction and postprocessing, in order to deal with the whole process of time series forecasting.

Keywords Time series forecasting · Machine learning · R

1 Introduction

Nowadays, a large number of research and society fields, such as financial [1], energy [7] and medical [20], are interested in the outstanding benefits that can be obtained from time series forecasting [6].

A time series can be defined as a time-ordered sequence of values provided by a source. In most cases, these values are taken at consecutive points in time. In statistics, the time series phenomenon is usually addressed from the theory of stochastic processes [8].

Traditionally, time series forecasting has been tackled with statistical methods, being the autoregressive integrated moving average (ARIMA) [3] among the most widely used techniques. Notwithstanding, machine learning (ML) algorithms [15] have been also used to predict these temporal future values in late years, even outperforming statistical methods in many cases. To carry out time series forecasting with ML, the first step is to transform the linear sequence of data into patterns/instances that conform a typical ML dataset. However, this step entails a lack in most ML software packages because they do not provide methods to accomplish this data adaptation task. Other typical data transformation actions used with statistical tools, such as de-trending, de-seasonalization or variance homogenization, have demonstrated their efficacy even with ML methods, but they are not available either in existing R packages for ML methods.

To fill these gaps, and in order to unify and ease time series forecasting specially with ML methods, we have developed **predtoolsTS**. This is an R package that provides a set of utilities covering all the steps involved in the prediction of new values from temporal data. The main contributions in **predtoolsTS** can be summarized as follows:

- It provides the methods needed to transform temporal data in order to apply ML methods to time series forecasting. This includes a preprocessing step at the beginning and a postprocessing one at the end. This functionality was not available until now for R users, so **predtoolsTS** fills an existing gap.
- Acting as a wrapper around several other R packages, **predtoolsTS** provides the practitioner with a fully automatic pipeline for performing time series forecasting. It is an easier way that carrying out each step manually, since the user has not the need to know the details of each package. Custom method implementations to fill the functionality gaps in these packages are also provided.

✉ Francisco Charte
fcharte@ujaen.es; francisco@fcharte.com

Alberto Vico
avm00016@red.ujaen.es

María D. Pérez-Godoy
lperez@ujaen.es

Antonio J. Rivera
arivera@ujaen.es

1 Dasci Andalusian Institute of Data Science and Computational Intelligence, Universidad de Jaén, Jaén, Spain

The **predtoolsTS** package is structured as four modules (preprocessing, modeling, predicting and postprocessing) in order to cover all the time series forecasting tasks. These aforementioned contributions are implemented as specific tools for:

- Carrying out exploratory data analysis and visualization.
- Applying different preprocessing methods to transform the time series data.
- Obtaining ML datasets from a linear sequence of temporal data.
- Using specific data partitioning techniques, such as *rolling window*, which are common in time series analysis.
- Accessing ARIMA methods and the automatic design of ARIMA-based models.
- Making accessible a large number of ML algorithms.
- Recovering the original temporal data by providing a postprocessing method for each available preprocessing one.

The rest of this paper is structured as follows. Background about time series forecasting, as well as existing R packages to accomplish this kind of task, is provided in Sect. 2. Section 3 outlines the structure of the **predtoolsTS** package, while details on how to use it are shown in Sect. 4. A few brief conclusions in Sect. 5 finalize the article.

2 Background

Since the proposed **predtoolsTS** package is aimed to ease time series forecasting, this section starts by providing a brief introduction to this field. Then, a succinct review on existing R packages related to this task is made.

2.1 Time series forecasting

In time series forecasting [4], given a sequence of temporal data $x_1, x_2, \ldots, x_n$ it is assumed that future values $x_{n+1}, \ldots, x_{n+2}$ can be obtained from past ones. Specifically, in a forecasting problem the x_{n+i} value, to be predicted, is called *horizon*, while known values used to predict this horizon $x_{n-j}, \ldots, x_{n-k}, \ldots x_{n-h}$, being j, k and h not necessarily consecutive intervals in the past, are called *lags*.

The first step when a forecasting problem is faced consists in carrying out an exploratory data analysis that characterize the time series data. Currently, it is accepted that there is a large number of forecasting methods performing better if the temporal data fulfill certain properties. Specifically, stationarity [5] is the main characteristic that researchers find in a time series. A process is said to be stationary when the distribution of the data does not change along time. This implies that the mean or the variance is kept constant on any sample taken from data. The property related to maintaining constant the variance is also known as homoscedasticity.

If a time series does not fulfill the stationarity trait it can be transformed. Examples of non-stationary time series are those with trend, seasonality, or an heterogeneous variance. In these cases, methods that imply de-trending, de-seasonalization and variance homogenization can be applied. The transformation of temporal data to achieve suitable characteristics constitutes the well-known preprocessing stage in any forecasting methodology. **predtoolsTS** provides utilities to carry out exploratory data analysis and data visualization, as well as the necessary preprocessing methods to transform the time series data in order to improve the performance of forecasting methods.

Time series analysis and forecasting have been faced by statistical methods for long time [6]. However, in late years ML methods have been used with successful results [15], even outperforming the classical methods in many cases. To predict future values, ML algorithms learn from data and obtain models that describe the series behavior. To conduct time series forecasting with a ML method, the first step consists in transforming the lineal sequence of data in patterns/instances that conforms a typical ML dataset.

To face time series forecasting through ML methods, it is necessary to obtain ML datasets from the temporal data. Each pattern of this dataset is composed by specific input values: the lags, and one output: the horizon. The determination of these lags depends on the followed methodology, being a well-known example the use of the autocorrelation function (ACF) integrated in **predtoolsTS**. Accomplishing the proper dataset structure is mandatory, but this functionality is not found in most ML packages for time series forecasting tasks. This is one of the outstanding possibilities allowed by **predtoolsTS**. Following with the dataset preparation phase, one of the peculiarities of the time series data field is how cross-validation is accomplished. By contrast to the typical cross-validation methodology used in classification, where patterns are randomly selected, in time series it is mandatory to keep the order of the instances. These data partitioning techniques are known as *rolling window* [17]. As above, current ML packages lack these partitioning techniques, which are available in **predtoolsTS**.

Once the dataset is adequately built, any ML regression method can be used to carry out time series forecasting. **predtoolsTS** makes possible accessing a large number of regression methods, specifically all those implemented in the widely used **caret** package [12].

Finally, in a postprocessing stage linked to the preprocessing previously applied, methods that restore the predicted values to the original ranges can be used. This is necessary to obtain the final predicted values. **predToolTS** provides a postprocessing method for each available preprocessing one.

Table 1 Main features in predtoolsTS and other time series forecasting R packages

Functionality	predtoolsTS	forecast	timeSeries	tseries
Data visualization	✓	✓	✓	✓
Automated data pre/postprocessing	✓	✗	✗	✗
Forecasting with ARIMA	✓	✓	✗	✓
Forecasting with data mining methods	✓	✗	✗	✗
Comparison of multiple models	✓	✗	✗	✗

2.2 Related packages

Classically, time series analysis has been addressed from the statistics area, and therefore R, the most used statistics framework, has the largest number of packages for dealing with these temporal data. In *CRAN Task View: Time Series Analysis*[1] a wide review about these packages is provided, with more than 250 tools. Some well-known classes and packages for time series prediction are briefly described below.

ts, included in the **stats** package [14], is the base class to represent a time series with its intrinsic characteristics such as order, period of time, etc. This class is used by most of the time series packages in R, and also by **predtoolsTS**. Other packages that implement classes to store, manage and calculate common statistical measures from chronological data are **zoo** [21], **xts** [16] and **timeSeries**.

Nowadays, **forecast** [10] is one of the most relevant packages to analyze and forecast time series. This package allows automatic forecasting using ARIMA [5], exponential smoothing, cubic splines and other statistical methods, as well as provides utilities for managing and visualizing temporal data.

Other forecasting packages that use common statistical techniques such as ARIMA, exponential smoothing or the theta method are **tseries** [18], **MAPA** [11], **forecTheta** [9], **gsarima** and **FitARMA** [13]. GARCH models [2] can be obtained from **tseries** or **fGarch**. Finally, forecast combination of statistical methods can be found in **ForecastComb** [19] or **forecastHybrid**.

In the previous software packages review, and as far as the authors are concerned, there is no R package that facilitates the use of ML methods applied to time series forecasting. **predtoolsTS** is a wrapper that unifies the use of statistical and ML methods for time series prediction. Thus, the package provides tools for transforming the linear representation of the temporal data into a dataset of patterns with the right structure for being processed by ML methods. It provides an uniform way of accessing the main statistical tools as well, along with a large numbers of ML regression methods.

[1] https://cran.r-project.org/web/views/TimeSeries.html.

Table 2 Methods in the **predtoolsTS** package that act as wrappers and the packages and functions they rely on

predtoolsTS method	Package	Function
prep.homogenize.boxcox	forecast	BoxCox
prep.check.acf	stats	acf
prep.check.adf	tseries	adf.test
modl.arima	forecast	auto.arima
pred.arima	stats	predict
postp.detrend.differencing	stats	diffinv
postp.deseason.differencing	stats	diffinv

Table 1 summarizes some of the salient functions provided by **predtoolsTS** and compared them with those present in the best-known time series forecasting R packages.

As will be explained below, **predtoolsTS** provides the user with abilities that were not available in existing time series forecasting packages. However, this package also offers an easier access to functions that were already present in some other R packages, mostly related to the use of classical time series processing approaches such as ARIMA. Table 2 indicates for each **predtoolsTS** method that acts as a wrapper[2] the package and function it relies on. Most of these methods are for internal use, so the user will never directly call them.

3 The predtoolsTS package

The major contribution of the **predtoolsTS** package is that it provides the user with an uniform interface to perform preprocessing, modeling, prediction and postprocessing of time series, regardless of the specific algorithms used to perform these tasks. The services offered by the package are grouped into four modules, each of which exports a main function. This section describes the basis of these four modules.

[2] The **predtoolsTS** package also relies on basic R functions and regression models in the **caret** package to accomplish its work.

3.1 Data preprocessing

The main function in this module is named **prep()**. Its goal is to transform an input time series, given as a **ts** object, into a stationary series, returned as a **prep** object.[3] This transformation is achieved after removing certain characteristics, such as seasonality or the presence of a trend, in order to improve efficiency of machine learning methods.

Using the parameters of the **prep()** function the user can choose among different deseasoning and de-trending methods. By default, the differencing method is applied to both processes. A variance homogenization method is used as well, in order to fulfill the homoscedasticity characteristic needed by the modeling function.

Before calling the **prep()** function, some previous checks can be performed by means of methods such as **prep.check.acf()** and **prep.check.adf()**, whose goal is helping the user to determine if the time series has seasonality factor or not. Individual preprocessing tasks can be executed as well, by calling the specific functions exported by this module.

3.2 Model generation

Once we have a stationary time series, having get rid of trends, seasonality and non-homogeneous variance, the data is ready to produce a predictive model. This is the goal of **modl()**, the main function in the model generation module. The input to this function can be a **ts** object or a **prep** object holding a preprocessed time series. The remainder arguments are all optional, being used to set the configuration of the model to be obtained.

The predictive model can be produced either by the statistical ARIMA method, which is the default option, or relying in any regression algorithm available in the **caret** package. To choose the second alternative the user has to set the **method** parameter to **"dataMining"**, stating with the **algorithm** parameter which specific regression method will be used. The remainder arguments of the **modl()** functions allow adjusting which intervals of the time series will be used to train and test the model.

3.3 Forecasting

At this stage the model obtained in the previous phase is used to forecast new values of the time series. For doing so, the **pred()** function is called providing it with the object returned by **modl()**. The **n.headed** parameter sets the number of values to be predicted, 20 by default. This method returns a list containing two **ts** objects as result, the original time series and the predicted one.

[3] Standard R methods, such as **summary()** and **plot()**, can be used with this object class.

The forecasting module also offers a function named **pred.compareModels()**. It is aimed to ease the comparison of several predictions, presumably obtained using different models. It takes as input the original time series and up to five more **ts** objects with predictions, plotting all of them together. Additional parameters aimed to set the legends in the plot can be provided.

3.4 Data postprocessing

If the data provided to **modl()** function have been preprocessed, aiming to improve the obtained model and hence the predictions, it have to be taken into account that those predictions will need some postprocessing. Calling the **postp()** function is mandatory in this specific case. It will apply the reverse transformations in order to restore time series values to their original ranges.

The **postp()** function takes two parameters, the predictions obtained in the forecasting step and the object previously returned by the **prep()** method. All the information needed by **postp()** to accomplish its work is held in this object.

4 How to use the predtoolsTS package

In this section, a few examples showing how to use the functions in the **predtoolsTS** package are provided. All of them use the **AirPassengers** dataset, a time series supplied by the **datasets** package which is included with the base R distribution. It is a monthly time series with the amount of international airline passengers going from 1949 to 1960.

```
> library("predtoolsTS")
> data("AirPassengers")
> plot(AirPassengers) # Explore the time
    series
> prep.check.acf(AirPassengers)
> prep.check.adf(AirPassengers)

Augmented Dickey--Fuller Test
data:  tserie
Dickey--Fuller = -7.3186, Lag order = 5, p-
    value = 0.01
alternative hypothesis: stationary
```

Example 1 Load the package and explore a time series

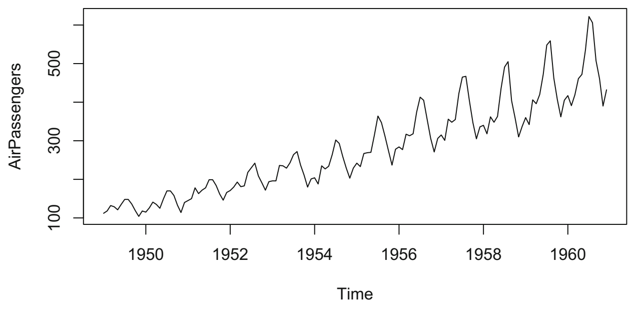

Fig. 1 **AirPassengers** time series

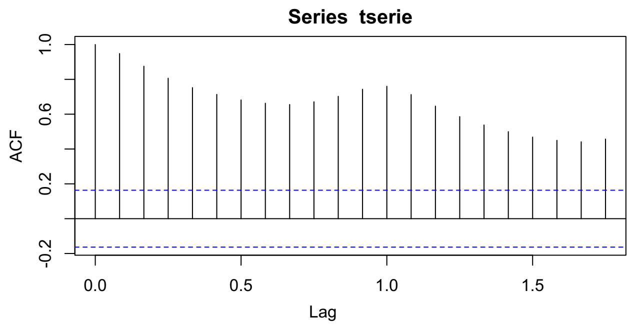

Fig. 2 Partial autocorrelation plot

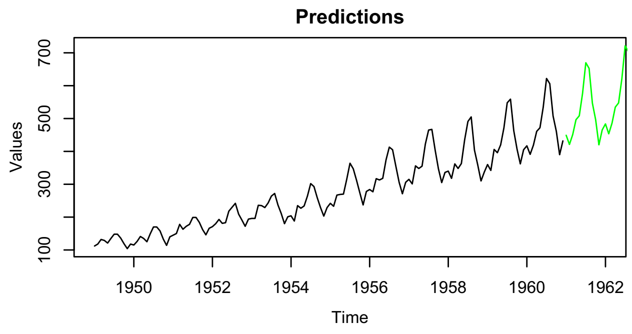

Fig. 3 The predictions are drawn in a different color

The first step would be the installation of the **predtoolsTS** package from CRAN, by issuing the usual **install.packages()** command. This will also install all the package's dependencies. Once the package is installed, it can be loaded and its functions called. We can start by loading and exploring the data we are going to work with. The plots (see Figs. 1, 2) and test results will show that **AirPassengers** is a time series having a clear tendency and also a seasonality factor.

Therefore, the following step would be preprocessing the data to transform it in a stationary time series. Then, a model would be generated and it would be used to obtain a set of predictions. These, along with the original data, are eventually plotted as shown in Fig. 3. The two code snippets provided below produce the same result.

```
    # Preprocess, model and predict
    prepAP <- prep(AirPassengers)
    modAP <- modl(prepAP)
    predAP <- pred(modAP)
    postAP <- postp(predAP, prepAP)
    plot(postAP)

    # Preprocess, model and predict with pipe
        operator
    library(magrittr)
    AirPassengers %>% #
    prep() %>% # Preprocessing
    modl() %>% # Modeling
    pred() %>% # Prediction
    postp(prep(AirPassengers)) %>%  #
        Postprocessing
    plot()     # Plotting
```

Example 2 Preprocess, model, predict and plot

As can be observed, the use of the `%>%` operator allows performing all the steps, preprocessing, model generation, prediction and postprocessing, in a very straightforward way. It works as a pipeline, where the original data is introduced, transformed, used to learn the model, etc.

In the above example, all methods use default parameters. The following one shows how to configure a model with an underlying data mining method, as well as how to compare

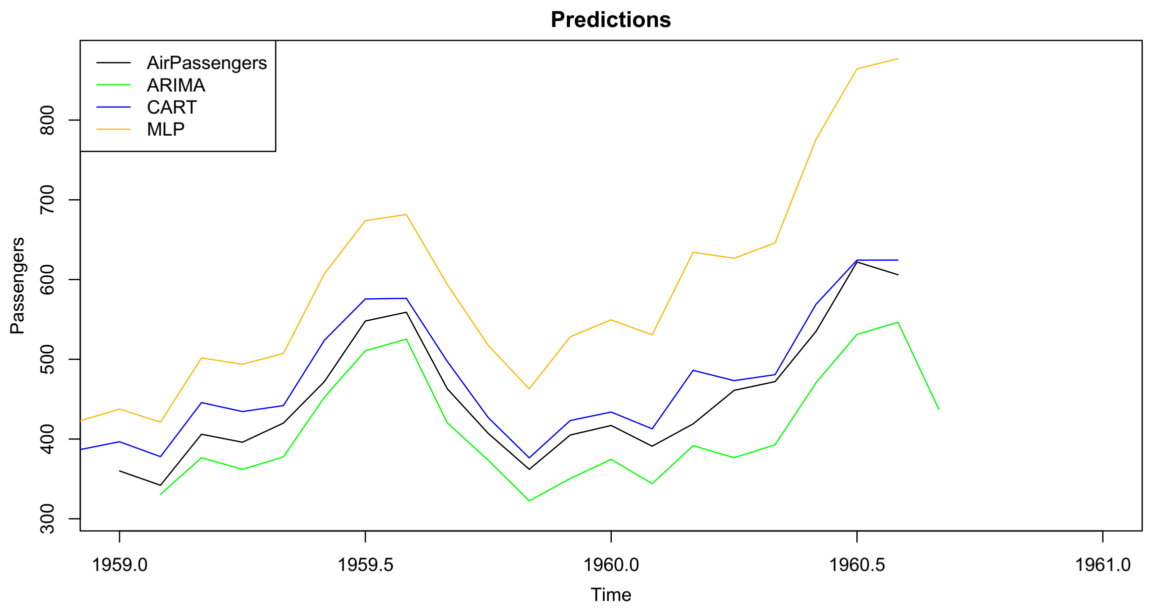

Fig. 4 Comparison of the three sets of predictions and the real data

the predictions returned by several models. The result would be similar to that shown in Fig. 4.

```
# Extract part of the time series to train
    the models
ap <- window(AirPassengers, start = 1949, end
     = 1959)
prepAP <- prep(ap)  # Preprocess the data

# Predictions obtained with default ARIMA
    model
predARIMA <- ap %>%
prep() %>%
modl() %>%
pred() %>%
postp(prepAP)

# Predictions produced by two data mining
    methods
predCART <- ap %>% prep() %>%
modl("dataMining", "rpart") %>%
pred() %>%
postp(prepAP)

predMLP <- ap %>% prep() %>%
modl("dataMining", "mlp") %>%
pred() %>%
postp(prepAP)

# Visual comparison of the three sets of
    predictions
# and the real data
pred.compareModels(
window(AirPassengers, 1959, c(1959,20)),
predARIMA$predictions, predCART$predictions,
predMLP$predictions,
legendNames=c('AirPassengers','ARIMA','CART',
    'MLP'),
yAxis = 'Passengers', legendPosition = '
    topleft')
```

Example 3 Compare three predictive models

5 Conclusions

In this paper **predtoolsTS**, a package for dealing with the whole task of time series forecasting, is presented. The package is structured into four modules for the preprocessing, modeling, prediction and postprocessing of temporal data. They offer a consistent and easy way to run statistical and ML methods for time series forecasting.

The main contributions in `predtoolsTS` are twofold. First, the addition of a new functionality to the R's package ecosystem that allows transforming time series data to the proper format required by standard ML methods. Second, providing the user with an homogeneous and streamline access to both statistical and ML algorithms for predicting data sequences.

Acknowledgements This work was partially supported by the project TIN2015-68854-R (FEDER Founds) of the Spanish Ministry of Economy and Competitiveness.

References

1. Barboza, F., Kimura, H., Altman, E.: Machine learning models and bankruptcy prediction. Expert Syst. Appl. **83**, 405–417 (2017)
2. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. J. Econom. **31**(3), 307–327 (1986)
3. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 4th edn. Wiley, Hoboken (2008)
4. Brockwell, P.J., Davis, R.A., Calder, M.V.: Introduction to Time Series and Forecasting, vol. 2. Springer, Berlin (2002)
5. Das, S.: Time Series Analysis. Princeton University Press, Princeton (1994)
6. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**(3), 443–473 (2006)
7. Deb, C., Zhang, F., Yang, J., Lee, S.E., Shah, K.W.: A review on time series forecasting techniques for building energy consumption. Renew. Sustain. Energy Rev. **74**, 902–924 (2017)
8. Doukhan, P.: Stochastic Models for Time Series, vol. 80. Springer, Berlin (2018)
9. Fiorucci, J.A., Louzada, F., Yiqi, B., Fiorucci, M.J.A.: Package 'forectheta' (2016)
10. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **26**(3), 1–22 (2008)
11. Kourentzes, N., Petropoulos, F., Trapero, J.R.: Improving forecasting by estimating time series structural components across multiple frequencies. Int. J. Forecast. **30**(2), 593 (2014)
12. Kuhn, M., et al.: Building predictive models in r using the caret package. J. Stat. Softw. **28**(5), 1–26 (2008)
13. McLeod, A.I., Zhang, Y.: Faster arma maximum likelihood estimation. Comput. Stat. Data Anal. **52**(4), 2166–2176 (2007)
14. McLeod, A.I., Yu, H., Mahdi, E.: Time series analysis with r. In: Subba Rao, T., Subba Rao, S., Rao, C.R. (eds.) Handbook of Statistics, vol. 30, pp. 661–712. Elsevier, Amsterdam (2012)
15. Naing, W.Y.N., Htike, Z.Z.: State of the art machine learning techniques for time series forecasting: a survey. Adv. Sci. Lett. **21**(11), 3574–3576 (2015)
16. Ryan, J.A., Ulrich, J.M.: xts: Extensible Time Series. R package version 0.8-2 (2011)
17. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. Int. J. Forecast. **16**(4), 437–450 (2000)
18. Trapletti, A., Hornik, K.: tseries: Time Series Analysis and Computational Finance (2018). R package version 0.10-46
19. Weiss, C.E., Raviv, E., Roetzer, G.: Forecast combinations in r using the forecastcomb package. R J. **10**(2), 262–281 (2018)
20. Zecchin, C., Facchinetti, A., Sparacino, G., Cobelli, C.: Jump neural network for online short-time prediction of blood glucose from continuous monitoring sensors and meal information. Comput. Methods Progr. Biomed. **113**(1), 144–152 (2014)
21. Zeileis, A., Grothendieck, G.: Zoo: S3 infrastructure for regular and irregular time series. J. Stat. Softw. **14**(6), 1–27 (2005)