Multivariate Regression Tree for Pattern-Based Forecasting Time Series with Multiple Seasonal Cycles

Grzegorz Dudek(✉)

Department of Electrical Engineering, Czestochowa University of Technology, Czestochowa, Poland
dudek@el.pcz.czest.pl

Abstract. Multivariate regression tree methodology is used for forecasting time series with multiple seasonal cycles. Unlike typical regression trees, which generate only one output, multivariate approach generates many outputs in the same time, which represent the forecasts for subsequent time-points. In the proposed approach a time series is represented by patterns of seasonal cycles, which simplifies the forecasting problem and allows the forecasting model to capture multiple seasonal cycles, trend and nonstationarity. In application example the proposed model is applied to forecasting electrical load of power system. Its performance is compared with some alternative models such as CART, ARIMA and exponential smoothing. Application examples confirm good properties of the model and its high accuracy.

Keywords: Multivariate regression tree · Seasonal time series forecasting · Pattern-based forecasting

1 Introduction

Multivariate or multi-output regression aims to simultaneously predict multiple output variables. There are two general approaches for solving multi-output regression problems: either by decomposing the problem into multiple single-output problems or, by adapting a model so that it directly handles multi-output data. The former solution can be time consuming task especially in the case of many outputs and large size of the training data. Moreover, the relationships among the output variables are ignored because they are predicted independently, which may affect the accuracy of the predictions. The latter solution is not as popular because of the complexity of the multivariate regression model. It should be able to capture not only the underlying relationships between input and output variables but also the internal relationships between output variables. According to past empirical works [1, 2] multi-output models ensure better predictive performance especially when the output variables are correlated.

The multivariate regression model learns from the training set $\Psi = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where $\mathbf{x} \in X = \mathbb{R}^n$ is a n-dimensional input vector and $\mathbf{y} \in Y = \mathbb{R}^m$ is a m-dimensional output vector, a target function f which assigns to each input vector an output vector: $f: X \rightarrow Y$. State-of-the-art multi-output regression methods are defined as extensions of standard single-output methods such as statistical methods, support vector machines, kernel methods, regression trees, and classification rules [3].

In the case of forecasting problems the input vector $\mathbf{x}$ includes predictors (past and present data) and the output vector $\mathbf{y}$ includes forecasted variables (future data). The output vector can contains variables of different types, which are dependent on the same predictors, e.g. demand for m various goods, or a single variable in different time-points, e.g. demand for a single good at time-points $t + 1, t + 2, \ldots, t + m$, where m is a forecast horizon. In the experimental part of this work we consider an example of short-term load forecasting, i.e. electrical hourly demand forecasting for the next day. Input vector includes power system hourly loads for the day preceding the forecasted day ($n = 24$) and output vector includes hourly loads for the next day ($m = 24$). Load time series exhibit multiple seasonal cycles of different lengths: daily, weekly and annual (Fig. 1). The daily and weekly profiles change during the year. The daily profile also depends on the day of the week. The load time series expresses trend and is nonstationary in mean and variance. To deal with these all features the load time series is preprocessed to filter out a trend, weekly and annual cycles. The way of time series representation is described in Sect. 2.

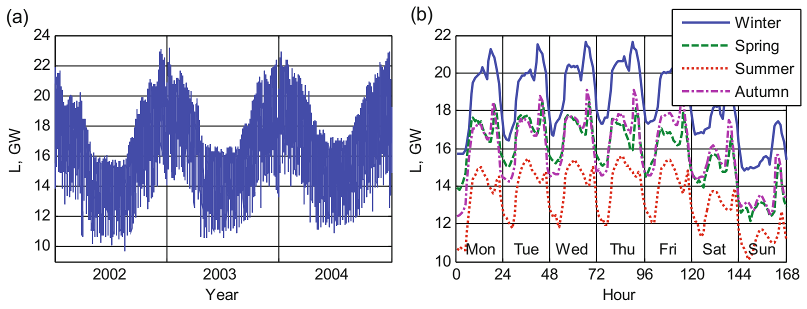

Fig. 1. The hourly electricity demand in Poland in three-year (a) and one-week (b) intervals.

Many various methods has been developed for load time series forecasting. They can be roughly classified as conventional and unconventional ones. Conventional approaches employ regression methods, smoothing techniques and statistical analysis such as ARIMA and exponential smoothing. Unconventional approaches use computational intelligence and machine learning methods such as: neural networks (NN), fuzzy inference systems, neuro-fuzzy systems and support vector machines. These approaches are reviewed in [4, 5]. Most of them are single-output models, but others are able to predict many outputs simultaneously. They include [6–8]: radial basis function NN, generalized regression NN, counterpropagation NN, self-organizing maps, artificial immune systems, nonparametric kernel regression, nearest neighbour regression and clustering-based models.

Regression trees as universal regression methods have been used for load time series forecasting as well [9]. They were originally developed as single-output models but they can be easily adapted to multi-output models. Advantages of multi-output regression trees (MRT) are: much smaller size than a set of single-output regression trees constructed for individual outputs, and taking into account the relationships between outputs when the tree is constructed. In this paper we define and explore MRT for power system load forecasting using patterns of daily cycles as inputs and outputs.

2 Time Series Representations Using Patterns

The daily periods of the load time series are preprocessed and are used as the input vectors for the forecasting model. They express daily shapes or profiles of the time series called x-patterns. The i-th daily period of the load time series $\mathbf{L}_i = [L_{i,1}\ L_{i,2}\ \ldots\ L_{i,n}]$ is represented by the input pattern $\mathbf{x}_i = [x_{i,1}\ x_{i,2} \ldots x_{i,n}] \in X = \mathbb{R}^n$. It is a normalized version of the load vector $\mathbf{L}_i$. The components of x-pattern are defined as follows [10]:

$$x_{i,t} = \frac{L_{i,t} - \bar{L}_i}{D_i} \tag{1}$$

where: $i = 1, 2, \ldots, N$ is the daily period number, $t = 1, 2, \ldots, n$ is the time series element number in the period i, $L_{i,t}$ is the t-th load time series element in the period i, $\bar{L}_i$ is the mean load in the period i, and $D_i = \sqrt{\sum_{l=1}^{n} (L_{i,l} - \bar{L}_i)^2}$ is the dispersion of the time series elements in the period i.

As normalized vectors x-patterns have unity length, zero mean and the same variance. Note that the load time series, which is nonstationary in mean and variance, is represented by x-patterns having the same mean and variance. The x-pattern carries information about the shape of the daily load curve. A trend and also weekly and annual variations are filtered.

An output vector $\mathbf{y}$ expresses the forecasted daily profile. When the forecast horizon is τ (in days), the forecasted load vector $\mathbf{L}_{i+\tau} = [L_{i+\tau,1}\ L_{i+\tau,2} \ldots L_{i+\tau,n}]$ is represented by an output pattern $\mathbf{y}_i = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,n}] \in Y = \mathbb{R}^n$. Their components are defined as follows:

$$y_{i,t} = \frac{L_{i+\tau,t} - \bar{L}_i}{D_i} \tag{2}$$

where: $i = 1, 2, \ldots, N$, $t = 1, 2, \ldots, n$.

Note that in (2) we use the coding variables $\bar{L}_i$ and D_i for the day i instead of the day $i + \tau$. This is because their values for the day $i + \tau$ are unknown at the moment of forecasting. Using their known values for the day i we can calculate the forecasted load value from transformed Eq. (2):

$$\widehat{L}_{i+\tau,t} = \widehat{y}_{i,t} D_i + \bar{L}_i \tag{3}$$

where $\widehat{y}_{i,t}$ is the t-th component of the y-pattern forecasted by the model.

Due to preprocessing daily periods of the load time series using x- and y-patterns we unify the input and output data and simplify relationships between them. This is further discussed in [10].

3 Multivariate Regression Trees for Forecasting

Multivariate regression trees is an extension of CART (Classification and Regression Trees [11]) proposed by Segal [12]. It works exactly the same way as classical CART, except that there is multiple response variables instead of one. As in CART, the response variables can be continuous or class symbols. Predictor variables can be of different types: quantitative or qualitative. Due to dealing with different types of data, decision trees are an attractive tool for classification and regression problems.

Tree-based methods partition the feature space X into a set of rectangles, and then fit a simple model, usually a constant, in each one. Let us consider multivariable regression problem with multi-input and multi-output continuous variables: $\mathbf{x}$ and $\mathbf{y}$, respectively (in our case $\mathbf{x}$ are n-dimensional input patterns and $\mathbf{y}$ are n-dimensional output patterns). MRT is constructed with a standard top-down induction algorithm. We are starting to build a tree splitting the X space into two regions: $x_j \leq t$ and $x_j > t$. This occurs in node 1 (root node). The variable x_j and split-point t are chosen to achieve the best fit. Then we model the response by the mean of $\mathbf{y}$ in each region. In the next step one or both of regions are split into two more regions in the nodes at the next level of the tree, and this process is continued, until some stopping rule is applied. The result is a partition into the K disjoint regions. The corresponding regression model predicts $\mathbf{y}$ with a constant $\widehat{\mathbf{y}}_k$ in region R_k:

$$\widehat{\mathbf{y}}_k = \frac{1}{N_k} \sum_{i:\mathbf{x}_i \in R_k} \mathbf{y}_i \tag{4}$$

This model is represented by the binary tree. The full dataset of N samples is split into two subsets in the root node. Samples satisfying the condition $x_j \leq t$ at each node are assigned to the left branch, and the others to the right branch. The terminal nodes (leaves) of the tree correspond to the final K regions.

The algorithm of growing a regression tree needs to automatically decide on the splitting variable x_j and split-point t at each node. We seek the splitting variable j among all n variables and split-point t with a greedy algorithm testing all variables and all possible split-points for them. To do so we define a function which is maximized during the searching process. It bases on within node sum of squares (it plays a role of the impurity measure of the node):

$$SS_k = \sum_{i:\mathbf{x}_i \in R_k} \sum_{l=1}^{n} (y_{i,l} - \bar{y}_l)^2 \tag{5}$$

where $\bar{y}_l$ is the average of l-th components of the samples in node k.

The splitting function expressing reduction in impurity after split is of the form:

$$\phi_k(j,t) = SS_k - SS_l - SS_m \tag{6}$$

and the best split maximizes this function:

$$\max_{\substack{j \in \{1,2,\ldots,n\} \\ t \in \Phi_j}} \phi_k(j,t) \tag{7}$$

where l and m are daughter nodes of the node k, which is split, and Φ_j denotes the set of all split-points possible for variable x_j.

The tree is constructed by recursively splitting nodes so as to maximize the above criterion. Tree size is a tuning parameter governing complexity of the model. The optimal tree size should be adaptively chosen from the data. We consider two approaches which determine the tree size. The first one splits tree nodes if the value of criterion (4) is positive (reduction of impurity) and the number of samples in the node is higher than L. This approach produces a tree with internal nodes having at least L samples and leaves having less than L samples. The second approach splits nodes if the impurity is reduced and the variance of y-patterns in the node is higher than v:

$$Var_k = \frac{1}{N_k n} \sum_{i:\,\mathbf{x}_i \in \Phi_k} \sum_{l=1}^{n} \left(y_{i,l} - \bar{y}_l\right)^2 > v \tag{8}$$

Parameters L and v determine the tree size. Higher values of these parameters leads to the smaller trees, which tend to underfitting. On the other hand, too small values lead to bigger trees and overfitting. The optimal values of L and v are estimated in the local version of leave-one-out cross-validation, which is presented in Fig. 2.

```
1. Find $k$ nearest neighbors of the query pattern $\mathbf{x}$ in the
   training set.
2. Build MRT using the training set excluding $i$-th nearest
   neighbor.
3. Calculate MRT error for $i$-th nearest neighbor.
4. Repeat steps 2 and 3 $k$ times for the next nearest neigh-
   bors.
5. Calculate average error over $k$ nearest neighbors as an
   estimate of generalization error.
```

Fig. 2. Algorithm of the local leave-one-out.

4 Application Example

In this section the proposed MRT model was applied to forecast the electricity load demand. As it was described in Sect. 1 electricity load time series exhibits multiple seasonal cycles. The data used for the experiments were retrieved from Polish power system. They contain hourly electricity load data from 2002 to 2004 (the load time series is shown in Fig. 1). The forecast horizon is $\tau = 1$, i.e. 24 h ahead. The MRT models are constructed for 31 days of January 2004 and 31days of July 2004 (in total 62 models for 62 forecasting tasks). For each forecasting task (test sample) the learning set is created individually from the historical data. It contains pairs of patterns representing the same days of the week (Monday,…, Sunday) as the query pattern **x** and forecasted pattern **y**.

Two variants of MRT models are used, which differ in the node splitting criterion:

- MRT1, where the node is split if the impurity is reduced and the number of samples in the node is higher than L,
- MRT2, where the node is split if the impurity is reduced and the variance in y-patterns in the node is higher than v.

The model parameters: number of samples L or variance v, are estimated in the local leave-one-out procedure, in which the validation samples are chosen one by one from the set of $k = 10$ nearest neighbors of the query pattern. In these procedures parameters were estimated using a grid search: L was searched from 4 to 50 with a step size of 2, and v was searched from 0.001 to 0.005 with a step size of 0.0002.

In Fig. 3 an example of MRT1 is shown: tree constructed for July 1, 2004. The optimal value of L was 32 in this case. As can be seen from this figure the tree has fifteen nodes in total, of which eight are leaves. The training y-patterns included in leaves are shown in Fig. 4. Note differences in shape between these patterns. They all represent Thursdays from history (from the period of January 1, 2002–June 30, 2004). Thick lines in this figure express average y-patterns, i.e. the tree response. Details of the tree in Table 1 are shown. There are only six variables used in the node tests: 1, 17, 19, 20, 21 and 22. They are the most important variables among 24 ones in partitioning the

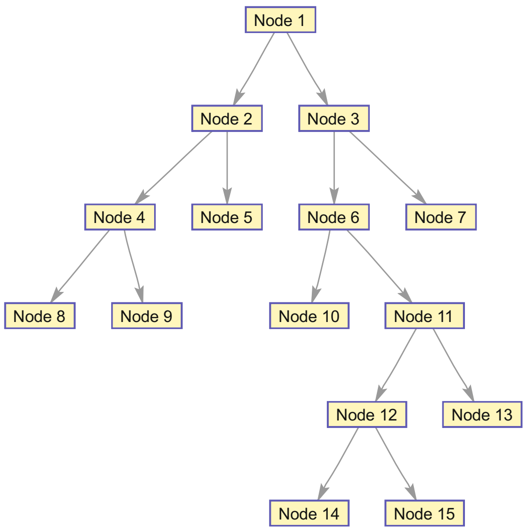

Fig. 3. MRT1 built for July 1, 2004.

input space X into regions. The largest reduction in impurity occurred in the root node (5.58), and also in node 3 (5.36). In the root node the set of 122 samples (the whole training set) was split into two subsets of 55 and 67 samples, respectively. In node 3 the largest reduction in impurity was achieved by excluding only one sample. As we can see form Fig. 4 this is an outlier sample. It represents January 2, 2002. It is atypical because the y-pattern for this day is encoded using the mean load $\bar{L}_i$ and dispersion D_i for the previous day (see (2)), which is New Year's Day. The lower mean load for New Year's Day causes rising of the y-pattern for the next day and consequently its atypical character.

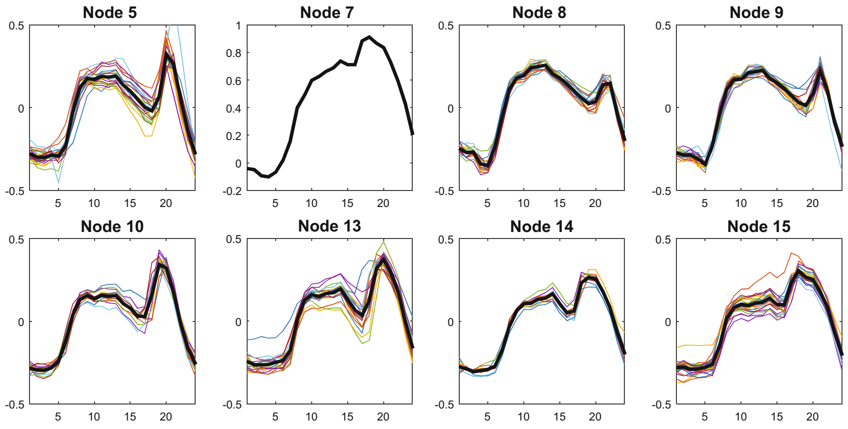

Fig. 4. Y-patterns in terminal nodes of MRT1 built for July 1, 2004; average y-patterns (tree response) with thick lines.

Trees constructed when using MRT2 strategy usually are bigger than trees constructed when using MRT1 strategy. This is shown in Fig. 5 presenting histograms of the number of nodes in the trees for these two cases. The average number of nodes in MRT1 is about 24 and in MRT2 about 36. For July 1, 2004 MRT2 built a tree having 49 nodes, of which 25 are leaves. Optimal value of v in this case was 0.0012. As in MTR1 the largest reduction in impurity (5.58) occurred in the root node, where the training set was split into two subsets of 55 and 67 samples. And also in node 3 (5.36), where the outlier representing January 2, 2002 was separated.

Variables selected for splitting in nodes are shown in Fig. 6. The most often selected variables are: 1, 7, 19 and 20, and the least often selected ones are: 15, 14 and 13.

For comparison the forecasts were performed using ARIMA, exponential smoothing, and classical regression tree (single-output CART). To find the optimal ARIMA and ES models the automated procedures were used implemented in the forecast package for the R system [13]. Distributions of the forecast errors (percentage errors PE) for the five models compared in Fig. 7 are shown. As we can see from this figure for the tree-based models the similar results were obtained. ARIMA and ES generate higher errors. The medians of PE were: 1.29 for ARIMA, 1.25 for ES, 1.01 for

Table 1. MRT1 details for July 1, 2004 (leaves in bold).

Node	Splitting variable j	Split-point t	Impurity reduction $\phi(j,t)$	#samples	Variance of y-patterns *Var*
1	19	0.1927	5.58	122	7.55E−03
2	20	0.1996	1.65	55	3.46E−03
3	1	−0.0686	5.36	67	7.43E−03
4	21	0.1684	0.21	34	1.37E−03
5	**–**	**–**	**–**	**21**	**3.57E−03**
6	22	0.0395	1.64	66	4.17E−03
7	–	–	–	1	0
8	**–**	**–**	**–**	**16**	**1.04E−03**
9	**–**	**–**	**–**	**18**	**1.17E−03**
10	**–**	**–**	**–**	**14**	**1.76E−03**
11	22	0.1284	1.40	52	3.50E−03
12	17	0.1364	0.29	34	1.84E−03
13	**–**	**–**	**–**	**18**	**3.41E−03**
14	**–**	**–**	**–**	**9**	**9.40E−04**
15	**–**	**–**	**–**	**25**	**1.68E−03**

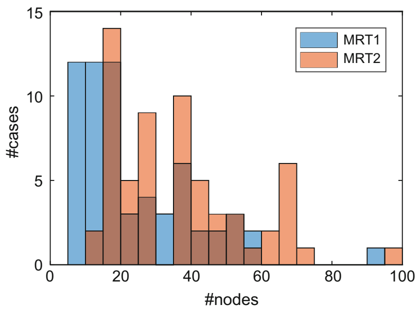

Fig. 5. Histograms of the number of nodes in the trees.

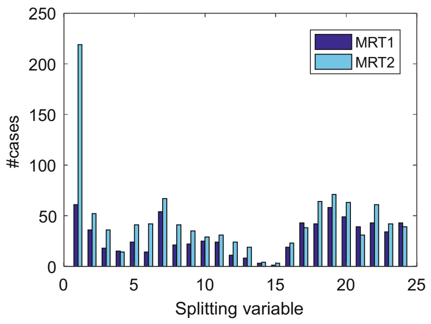

Fig. 6. Splitting variables in the trees.

regression tree, 1.04 for MRT1, and 1.09 for MRT2. The Wilcoxon signed-rank tests indicated that there is no statistically significant difference in errors between regression tree, MRT1 and MRT2. But there is a significant difference in errors between ARIMA/ES and each of tree-based models.

The proposed MRTs generate multi-output response, keeping the relationships between the output variables (y-pattern components). In the case of single-output models, like classical CART, these relationships are ignored because variables are

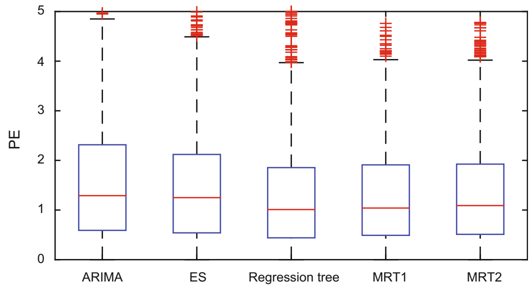

Fig. 7. Boxplots for percentage errors generated by the models.

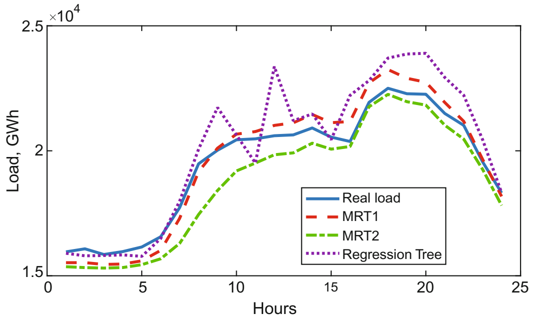

Fig. 8. Real load and forecasts for January 5, 2004.

predicted independently. This may cause lack of smoothness in the forecasted curve. Such example in Fig. 8 is illustrated. For January 5, 2004, in the case of regression tree model we can observe zigzag effect in the middle part of the curve.

5 Conclusion

The proposed model based on multivariate regression trees generates forecasts of the time series many steps ahead in the same time. The model is constructed taking into account not only the underlying relationships between input and output variables but also the internal relationships between output variables. This can lead to improvement in predictive performance especially when the output variables are correlated. The multi-output tree is much smaller than a set of single-output regression trees constructed for individual outputs. So, the learning process is easier and less time consuming. Another advantage of multivariate approach is that the zigzag problem, which appears when single-output model is used for forecasting output variables independently, is eliminated.

In the proposed approach a time series is represented by patterns of seasonal cycles, which simplifies the forecasting problem and allows the model to capture multiple seasonal cycles, trend and nonstationarity. Additional time series transformations such as decomposition, detrending or differencing are not needed.

In the application example the proposed model was applied to forecasting electrical load of power system. It provided as accurate forecasts as classical regression tree and outperformed ARIMA and exponential smoothing models.

References

1. Breiman, L., Friedman, J.H.: Predicting multivariate responses in multiple linear regression. J. R. Stat. Soc. Ser. B **59**(1), 3–54 (1997)
2. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. **6**, 615–637 (2005)
3. Borchani, H., Varando, G., Bielza, C., Larrañaga, P.: A survey on multi-output regression. Wiley Interdisciplinary Rev. Data Mining Knowl. Discov. **5**(5), 216–233 (2015)
4. Tzafestas, S., Tzafestas, E.: Computational intelligence techniques for short-term electric load forecasting. J. Intell. Robot. Syst. **31**, 7–68 (2001)
5. Metaxiotis, K., Kagiannas, A., Askounis, D., Psarras, J.: Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher. Energy Convers. Manage. **44**, 1525–1534 (2003)
6. Dudek, G.: Neural networks for pattern-based short-term load forecasting: a comparative study. Neurocomputing **2015**, 64–74 (2016)
7. Dudek, G.: Pattern similarity-based methods for short-term load forecasting – Part 2: models. Appl. Soft Comput. **36**, 422–441 (2015)
8. Dudek, G.: Artificial immune system with local feature selection for short-term load forecasting. IEEE Trans. Evol. Comput. **21**(1), 116–130 (2017)
9. Dudek, G.: Short-term load forecasting using random forests. In: Filev, D., et al. (eds.) Intelligent Systems 2014, Advances in Intelligent Systems and Computing, vol. 323, pp. 821–828 (2015)
10. Dudek, G.: Pattern similarity-based methods for short-term load forecasting – Part 1: principles. Appl. Soft Comput. **37**, 277–287 (2015)
11. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)
12. Segal, M.R.: Tree-structured methods for longitudinal data. J. Am. Stat. Assoc. **87**(418), 407–418 (1992)
13. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **27**(3), 1–22 (2008)