

# PSF: Introduction to R Package for Pattern Sequence Based Forecasting Algorithm

by Neeraj Bokde, Gualberto Asencio-Cortés, Francisco Martínez-Álvarez and Kishore Kulat

**Abstract** This paper introduces the R package that implements the Pattern Sequence based Forecasting (PSF) algorithm, which was developed for univariate time series forecasting. This algorithm has been successfully applied to many different fields. The PSF algorithm consists of two major parts: clustering and prediction. The clustering part includes selection of the optimum number of clusters. It labels time series data with reference to such clusters. The prediction part includes functions like optimum window size selection for specific patterns and prediction of future values with reference to past pattern sequences. The PSF package consists of various functions to implement the PSF algorithm. It also contains a function which automates all other functions to obtain optimized prediction results. The aim of this package is to promote the PSF algorithm and to ease its usage with minimum efforts. This paper describes all the functions in the PSF package with their syntax. It also provides a simple example. Finally, the usefulness of this package is discussed by comparing it to `auto.arima` and `ets`, well-known time series forecasting functions available on CRAN repository.

## Introduction

PSF stands for Pattern Sequence Forecasting algorithm. PSF is a successful forecasting technique based on the assumption that there exist pattern sequences in the target time series data. For the first time, it was proposed in Martínez-Álvarez et al. (2008), and an improved version was discussed in Martínez-Álvarez et al. (2011a).

Martínez-Álvarez et al. (2011a) improved the label based forecasting (LBF) algorithm proposed in Martínez-Álvarez et al. (2008) to forecast the electricity price and compared it to other available forecasting algorithms such as ANN (Catalão et al., 2007), ARIMA (Conejo et al., 2005), mixed models (García-Martos et al., 2007) and WNN (Troncoso et al., 2007). These comparisons concluded that the PSF algorithm is able to outperform all these forecasting algorithms, at least in the electricity price/demand context.

Many authors have proposed improvements for PSF. In particular, Jin et al. (2014) highlighted the PSF algorithm limitations and suggested minute modification to minimize the computation delay. They suggested that, instead of using multiple indexes, a single index could make computation simpler and they used the Davies Bouldin index to obtain the optimum number of clusters.

Majidpour et al. (2014) compared PSF to kNN and ARIMA, and observed that PSF can be used for electric vehicle charging energy consumption. It also proposed three modifications in the existing PSF algorithm. First, instead of taking average of all the matched template, it only uses the most recent matched template. Second, if no match was found in the training data (which is possible), MPSF outputs the cluster center of the largest cluster as output. Third, instead of finding the optimum number of clusters starting at two, it starts  $k$  from 10% of the total number of samples to avoid degenerate clusters.

Shen et al. (2013) proposed an ensemble of PSF and five variants of the same algorithm, showing better joint performance when applied to electricity-related time series data.

Koprinska et al. (2013) attempted to propose a new algorithm for electricity demand forecast, which is a combination of PSF and Neural networks (NN) algorithms. The results concluded that PSF-NN is performing better than original PSF algorithm.

Fujimoto and Hayashi (2012) modified the clustering method in PSF algorithm. It used a cluster method based on non-negative tensor factorization instead of k-means technique and forecasted energy demand using photovoltaic energy records.

Martínez-Álvarez et al. (2011b) also modified the original PSF algorithm to be used for outlier occurrence forecasting in time series. The metaheuristic searches for motifs or pattern sequences preceding certain data, marked as anomalous in the training set. Then, outlying and regular data are separately processed. Once again, this version was shown to be useful in electricity prices and demand.

The analysis of these works highlights the fact that the PSF algorithm can be applied to many different fields for time series forecasting, outperforming many existing methods. Since the PSF algorithm consists of many dependent functions and the authors did not originally publish their code, the package here described aims at making PSF handy and with minimum efforts for coding.

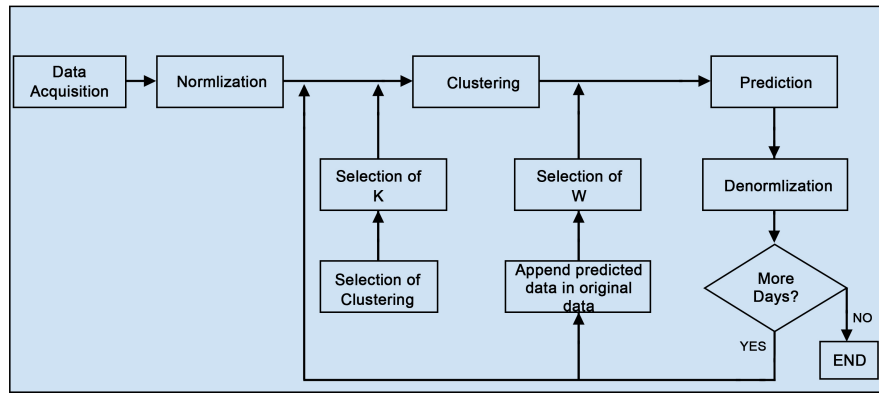
The rest of the paper is structured as follows. Section 2 provides an overview of the PSF algorithm.

Section 3 explains how the package in R has been developed. Illustrative examples are described in Section 4. Finally, the conclusions drawn from this work are summarized in Section 5.

## Brief description of PSF

The PSF algorithm consists of many processes, which can be divided, broadly, in two steps. The first step is clustering of data and the second step is forecasting based on clustered data in earlier step. The block diagram of PSF algorithm shown in Figure 1 was proposed by Martínez-Álvarez et al. (2008). The PSF algorithm is a closed loop process, hence it adds an advantage since it can attempt to predict the future values up to long duration by appending earlier forecasted value to existing original time series data. The block diagram for PSF algorithm shows the close loop feedback characteristics of the algorithm. Although there are various strategies for multiple-step ahead forecasting as described by Bontempi et al. (2013), this strategy turned out to be particularly suitable for electricity prices and demand forecasting, as the original work discussed.

Another interesting feature lies in its ability to simultaneously forecast multiple values, that is, it deals with arbitrary lengths for the horizon of prediction. However, it must be noted that this algorithm is particularly developed to forecast time series exhibiting some patterns in the historical data, such as weather, electricity load or solar radiation, just to mention few examples of usage in reviewed literature. The application of PSF to time series without such kind of inherent patterns might lead to the generation of not particularly competitive results.



**Figure 1:** Block diagram of PSF algorithm methodology.

The clustering part consists of various tasks, including data normalization, optimum number of clusters selection and application of k-means clustering itself. The ultimate goal of this step is to discover clusters of time series data and label them accordingly.

Normalization is one of the essential processes in any time series data processing technique. Normalization is used to scale data. The algorithm (Martínez-Álvarez et al., 2011a) used the following transformation to normalize the data.

$$X_j = \frac{X_j}{\frac{1}{N} \sum_{i=1}^N X_i} \quad (1)$$

where  $X_j$  is the input time series data and  $N$  is the total length of the time series.

However, the original PSF original algorithm used  $N = 24$  hours instead of  $N$  as the total length of the time series. This presents the problem of knowing all hours for each day to assess the mean. For such reason, the original formula has been replaced in this implementation by the standard feature scaling formula (also called unity-based normalization), which bring all values into the range  $[0, 1]$  Dodge (2003):

$$X'_j = \frac{X_j - \min(X_i)}{\max(X_i) - \min(X_i)} \quad (2)$$

where  $X'_j$  denotes the normalized value for  $X_j$ , and  $i = 1, \dots, N$ .

The reference articles (Martínez-Álvarez et al., 2008) and (Martínez-Álvarez et al., 2011a) used the k-means clustering technique to assign labels to sets of consecutive values. In the original manuscript, since daily energy was analyzed, clustering was applied to every 24 consecutive values, that is, to

every day. The advantage of k-means is its simplicity, but it requires the number of clusters as input. In Martínez-Álvarez et al. (2008), the Silhouette index was used to decide the optimum numbers of clusters, whereas in the improved version (Martínez-Álvarez et al., 2011a), three different indexes were used, which include the Silhouette index (Kaufman and Rousseeuw, 2008), the Dunn index (Dunn, 1974) and the Davies Bouldin index (Davies and Bouldin, 1979). However, the number of groups suggested by each of these indexes is not necessarily the same. Hence, it was suggested to select the optimum number of clusters by combining more than one index, thus proposing a majority voting system.

As output of the clustering process, the original time series data is converted into a series of labels, which is used as input in the prediction block of the second phase of the PSF algorithm. The prediction technique consists of window size selection, searching for pattern sequences and estimation processes.

Let  $x(t)$  be the vector of time series data such that  $x(t) = [x_1(t), x_2(t), \dots, x_N(t)]$ . After clustering and labeling, the vector is converted to  $y(t) = [L_1, L_2, \dots, L_N]$ , where  $L_i$  are labels identifying the cluster centers to which data in vector  $x(t)$  belongs to. Note that every  $x_i(t)$  can be of arbitrary length and must be adjusted to the pattern sequence existing in every time series. For instance, in the original work,  $x(i)$  was composed of 24 values, representing daily patterns.

Then the searching process includes the last  $W$  labels from  $y(t)$  and it searches for these labels in  $y(t)$ . If this sequence of last  $W$  labels is not found in  $y(t)$ , then the search process is repeated for last  $(W-1)$  labels. In PSF, the length of this label sequence is named as window size. Therefore, the window size can vary from  $W$  to 1, although it is not usual that this event occurs. The selection of the optimum window size is very critical and important to make accurate predictions. The optimum window size selection is done in such a way that the forecasting error is minimized during the training process. Mathematically, the error function to be minimized is:

$$\sum_{t \in TS} \|\bar{X}(t) - X(t)\| \quad (3)$$

where  $\bar{X}(t)$  are predicted values and  $X(t)$  are original values of time series data. In practice, the window size selection is done with cross validation. All possible window sizes are tested on sample data and corresponding prediction errors are compared. The window size with minimum error considered as the optimum window size for prediction.

Once the optimum window size is obtained, the pattern sequence available in the window is searched for in  $y(t)$  and the label present just after each discovered sequence is noted in a new vector, called ES. Finally, the future time series value is predicted by averaging the values in vector ES as given below:

$$\bar{X}(t) = \frac{1}{\text{size}(ES)} \times \sum_{j=1}^{\text{size}(ES)} ES(j) \quad (4)$$

where,  $\text{size}(ES)$  is the length of vector ES. The procedure of prediction in the PSF algorithm is described in Figure 2. Note that the average is calculated with real values and not with labels.

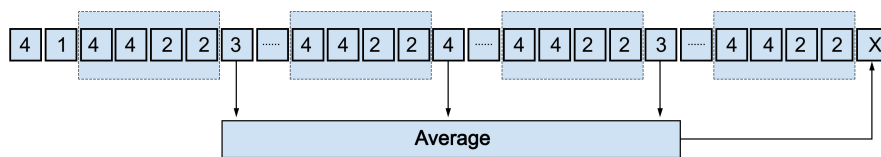


Figure 2: Prediction with PSF algorithm.

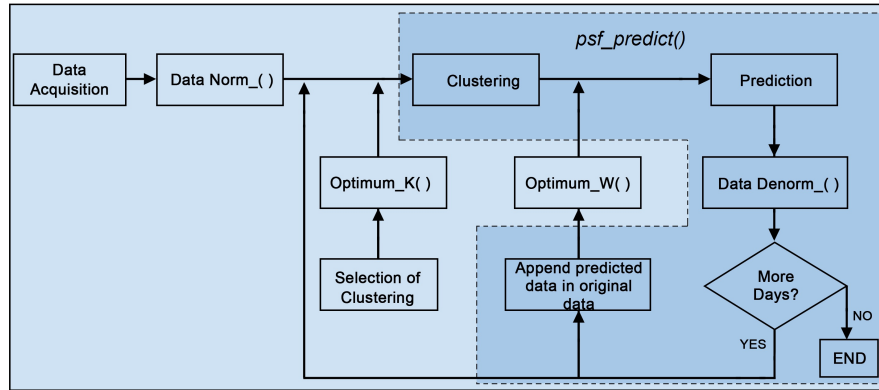
The algorithm can predict the future value for next interval of the time series. But, to make it applicable for long term prediction, the predicted short term values will get linked with original data and the whole procedure will be carried out till the desired length of time series prediction is obtained.

## PSF package description

This section introduces the PSF package in R language. Source code along with its reference manual can be found at <https://cran.r-project.org/package=PSF>. With reference to dependencies, this package imports package **cluster** (Maechler et al., 2015) and suggests packages **knitr** and **rmarkdown**. The package also needs the **data.table** package to improve data wrangling. PSF package consists of various functions. These functions are designed such that these can replace the block diagrams of

methodology used in PSF. The block diagram in Figure 3 represents the methodology mentioned in Figure 1 with the replacement of equivalent functions used in the proposed package.

The various tasks including the optimum cluster size, window size selection, pattern searching and prediction processes are performed in the package with different functions including `psf()`, `predict.psf()`, `plot.psf()`, `optimum_k()`, `optimum_w()`, `psf_predict()` and `convert_datatype()`. All functions in the PSF package version 0.4 onwards are made private except for `psf()`, `predict.psf()` and `plot.psf()` functions, so that users could not use it directly. Moreover, if users need to change or modify clustering techniques or procedures in private functions, the code is available at GitHub (<https://github.com/neerajdhanraj/PSF>).



**Figure 3:** Block diagram with PSF package functions replacement for PSF algorithm.

This section discusses each of these functions and their functionalities along with simple examples. All functions in the PSF package are loaded with:

```
library(PSF)
```

### Optimum cluster size selection

As aforementioned, clustering of data is one of the initial phases in PSF. The reference articles (Martínez-Álvarez et al., 2008) and (Martínez-Álvarez et al., 2011a) chose the k-means clustering technique for generating data clusters according to the time series data properties. The limitation of k-means clustering technique is that the adequate number of clusters must be provided by users. Hence, to avoid such situations, the proposed package contains a function `optimum_k()` which calculates the optimum value for the number of clusters, ( $k$ ), according to the Silhouette index. This function generates the optimum cluster size as output. In Martínez-Álvarez et al. (2011a), multiple indexes (Silhouette index, Dunn index and the Davies–Bouldin index) were considered to determine the optimum number of clusters. For the sake of simplicity and to save the calculation time, only the Silhouette index is considered in this package, as suggested in Martínez-Álvarez et al. (2008).

This `optimum_k()` is a private function which is not directly accessible by users. This function takes data as input time series, in any format be it matrix, data frame, list, time series or vector. Note that the input data type must be strictly numeric. This function returns the optimum value of number of clusters ( $k$ ) in numeric format.

### Optimum window size selection

Once data clustering is performed, the optimum window size needs to be determined. This is an important but tedious and time consuming process, if it is manually done. In Martínez-Álvarez et al. (2008) and Martínez-Álvarez et al. (2011a), the selection of optimum window size is done through cross-validation, in which data is partitioned into two subsets. One subset is used for analysis and the other one for validating the analysis. Since the window size will always be dependent on the pattern of the experimental data, it is necessary to determine the optimum window size for every time series data.

In the PSF package, the optimum window size selection is done with the function `optimum_w()`, which takes as input the time series data, the previous estimated  $k$  value, a set candidate  $w$  values to search in and the cycle of the input time series. This function estimates the optimum value for the window size such that the error between predicted and actual values is minimum. Internally, this

function divides the input time series data into two sub-parts. One of them is the last cycle data values which will be taken as reference to compare to the predicted values and to calculate RMSE values. The other sub-part is the remaining data set which is used as training part of the data set. The predicted values for the window size with minimum RMSE value is then taken as optimum window size. If more than one window size values are obtained with the same RMSE values, the maximum window size is preferred by the function `optimum_w()`. Like the `optimum_k()` function, the `optimum_w()` function is also a private function and users cannot directly access it.

### Prediction with PSF

The PSF package exposes three functions to the user. The first one, `psf()`, can build a PSF model from an univariate time series. The value returned by such function is a S3 object of class 'psf', which contains both original and normalized input time series along with other internal model adjustments. Once the PSF model is trained and returned, the user can invoke the S3 method `predict()` over the the 'psf' object specifying the desired number of forecasted values via the `n.ahead` parameter. This method returns a numeric vector. Finally, the third function exposed is the S3 method `plot()`, that produces a plot including both actual and predicted values from a PSF model and a numeric vector of predictions.

Internally, the prediction procedure is composed of data processing, optimum window size and cluster size selection. The prediction is done with the private function `psf_predict()`, which takes time series data, window size (`w`), cluster size (`k`) and an integer `n.ahead`. The value of `n.ahead` indicates the count up to what extend the forecasting is to be done by function `psf_predict()`. The time series data taken as input can be in any format supported by R language, be it time series, vector, matrix, list or data frame. The PSF package automatically converts it in suitable format using the private function `convert_datatype()` and proceeds further. All other input parameters `w`, `k` and `n.ahead` must be in integer format.

The function `psf_predict()` initiates the process of data normalization. Then, it selects the last `w` labels from the input time series data and searches for that number sequence in the entire data set. Additionally, it captures the very next integer value after each sequence and calculates the average of these values. The obtained averaged value along with denormalization is considered as raw predicted value. If the input parameter `n.ahead` is greater than unity, then the predicted value is appended to the original input data, and the procedure is repeated `n.ahead` times. Finally, the processed data is denormalized and returns a time series which replaces labels by the predicted values.

The `psf_predict()` function is also one of the private functions which takes input the dataset in `data.table` format, as well as another integer inputs including number of clusters (`k`), window size (`w`), horizon of prediction (`n.ahead`), and the `cycle` parameter, which discovers the cycle pattern followed by dataset. This function considers all input parameters and searches for the desired pattern in training data. While performing the searching process, if the desired pattern is obtained once or more times, it calculates the average of very next values for each repeated desired patterns in the whole dataset. Finally, this averaged value is considered as next predicted value and is appended to input time series data.

The `psf()` function uses `optimum_k()` and `optimum_w()` functions. The latter, in turn, uses the described `psf_predict()`. The function `psf()` returns the PSF trained model (S3 object of class 'psf') whose contents are described later in this section. The syntax for `psf()` function is shown below:

```
psf(data, k = seq(2, 10), w = seq(1, 10), cycle = 24)
```

Within the indicated syntax, the parameter `data` is an univariate time series in any format, e.g. time series, vector, matrix, list or data frame. Also, `data` must be strictly provided in numeric format. The parameter `k` is the number of clusters, whereas parameter `w` is the window size. Finally, the `cycle` parameter is the number of values that confirms a cycle in the time series. Usual values for the `cycle` parameter can be 24 hours per day, 12 months per year or so on and it is used only when input data is not in the time series format. If input data is given in time series format, the `cycle` is automatically determined by its internal frequency attribute.

If the user provides a single value for either `k` or `w`, then this value is used in the PSF algorithm and the search for their optimum is skipped. Furthermore, if the user provides a vector of values for either parameter `k` or `w`, then the search for optimum is limited to these values.

This `psf()` function returns a S3 object of class 'psf' which includes the 7 elements described below:

**original\_series** Original time series stored to be used internally to build further plots.

**train\_data** Adapted and normalized internal time series used to train the PSF model.

**k** The number of clusters optimized and used to train the model.  
**w** The window size optimized and used to train the model.  
**cycle** Determined cycle for the input time series.  
**dmin** Minimum value of the input time series (used to denormalize internally in further predictions).  
**dmax** Maximum value of the input time series (used to denormalize internally in further predictions).

The `psf()` function initiates the conversion of any type of data format into *data.table* which is an internal format for this function (this conversion is carried out by the private function `convert_datatype`). Secondly, it is checked whether the input dataset is multiple of `cycle` parameter or not. If not, it generates a warning to state that the dataset pattern is not suitable for further study. But if the dataset is multiple of the `cycle` value, the normalization of the dataset is carried out and subsequently reshaped according to the cycle of time series.

This reshaped dataset is then provided to `psf_predict()` function with other parameters including cluster size (`k`), window size (`w`), `n.ahead` and `cycle`, as mentioned before. In reply to this, the `psf_predict()` function returns a vector of future predicted values. Since the input dataset had been previously normalized, it is required to denormalize the predicted values. This is carried out by the `S3` method `predict.psf()`. This function returns a numeric vector with the denormalized predicted values.

### Plot function for PSF

The `plot.psf()` function allows users to plot both the actual values of a series and predicted values obtained by a PSF model. This function takes the trained PSF model, denoted by the first parameter named `x`, which includes internally the original time series, obtained by `psf()`, along with the predicted values obtained through `predict.psf()`, and optionally other plot describing variables, like plot title, legends, etc. This function generates the plot showing original time series data and predicted data with significant color changes. The plot obtained by `plot.psf()` possesses dynamic margin size such that it can include the input data set and all predicted values as per requirements. The syntax of `plot.psf()` function is as shown below. The usage of `plot.psf()` function is further discussed in the next section.

```
plot.psf(x, predictions, cycle = 24, ...)
```

### Example

This section presents the examples to introduce the use of the PSF package and to compare it with `auto.arima()` and `ets()` functions, which are well accepted functions in the R community working over time series forecasting techniques. The data used in this example are `nottem` and `sunspots` which are standard time series datasets available in R. The `nottem` dataset is the average air temperatures at Nottingham Castle in degrees Fahrenheit, collected for 20 years, on monthly basis. Similarly, `sunspots` dataset is mean relative sunspot numbers from 1749 to 1983, measured on monthly basis.

For both datasets, all the recorded values except for the final year are considered as training data, and the last year is used for testing purposes. The predicted values for final year for both datasets are discussed in this section.

In the following examples, model training, forecasting and plotting were shown for the dataset `nottem`, but the procedure is the same for the dataset `sunspots`. In first place, the model must be trained using the PSF package. as is shown below.

```
library(PSF)
nottem_model <- psf(nottem)
nottem_model

## $original_series
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1920 40.6 40.8 44.4 46.7 54.1 58.5 57.7 56.4 54.3 50.5 42.9 39.8
## 1921 44.2 39.8 45.1 47.0 54.1 58.7 66.3 59.9 57.0 54.2 39.7 42.8
## ...
##
## $train_data
##      V1      V2      V3      V4      V5      V6      V7
## 1: 0.26420455 0.2698864 0.3721591 0.4375000 0.6477273 0.7727273 0.7500000
```

```
## 2: 0.36647727 0.2414773 0.3920455 0.4460227 0.6477273 0.7784091 0.9943182
## ...
##
## $k
## [1] 2
##
## $w
## [1] 1
##
## $cycle
## [1] 12
##
## $dmin
## [1] 31.3
##
## $dmax
## [1] 66.5
##
## attr("class")
## [1] "psf"
```

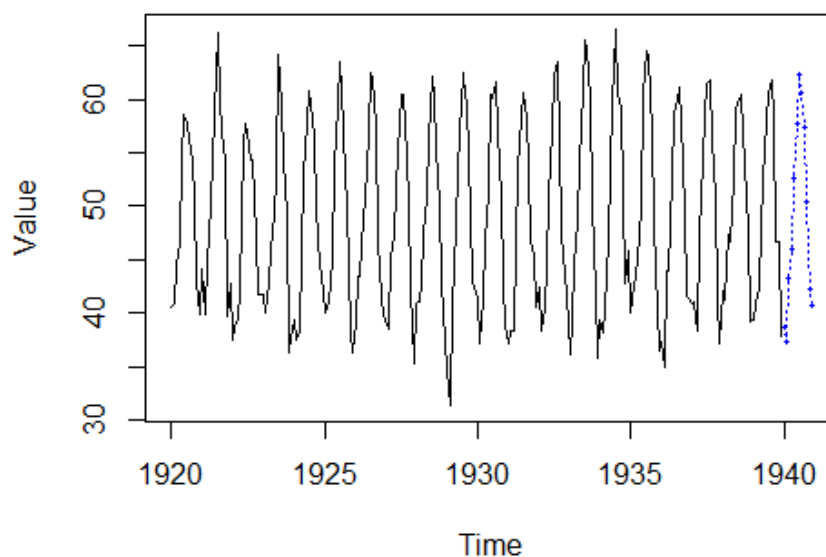
Once the model is trained, forecasted values for the time series can be obtained using the S3 method `predict()` function, as is shown below.

```
nottem_preds <- predict(nottem_model, n.ahead = 12)
nottem_preds

## [1] 38.97692 38.71538 42.49231 46.32308 52.91538 57.97692 61.87692
## [8] 60.19231 57.03846 49.42308 43.23846 40.21538
```

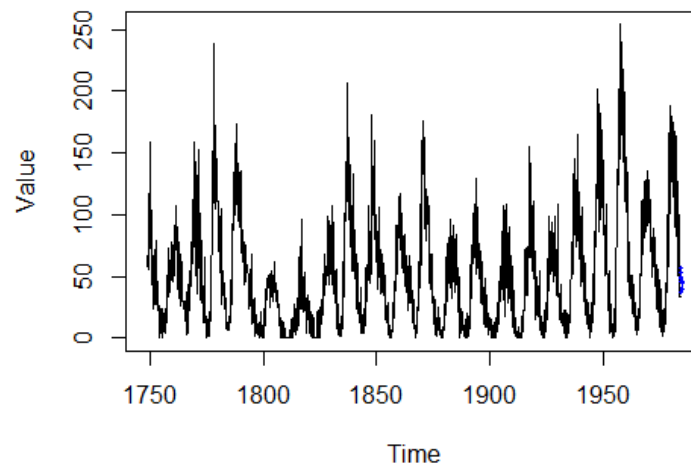
To represent the prediction performance in plot format, the S3 method `plot()` can be used, as shown in the following code.

```
plot(nottem_model, nottem_preds)
```



**Figure 4:** Plot showing forecasted results with function `plot.psf()` for nottem dataset.

Figures 4 and 5 show the prediction with the PSF algorithm for nottem and sunspots datasets, respectively. Such results are compared to those of `auto.arima()` and `ets()` functions from **forecast** R



**Figure 5:** Plot showing forecasted results with function `plot.psf()` for sunspots dataset.

Functions	<code>psf()</code>	<code>auto.arima()</code>	<code>ets()</code>
RMSE	2.077547	2.340092	32.78256

**Table 1:** Comparison of time series prediction methods with respect to RMSE values for nottem dataset.

package. `auto.arima()` function compares either AIC, AICc or BIC value and suggests best ARIMA or SARIMA models for a given time series data. Analogously, the `ets()` function considers the best exponential smoothing state space model automatically and predicts future values.

Tables 1 and 2 show comparisons for `psf()`, `auto.arima()` and `ets()` functions when using the Root Mean Square Error (RMSE) parameter as metric, for nottem and sunspots datasets, respectively. In order to avail more accurate and robust comparison results, error values are calculated for 10 times and the mean value of error values for methods under comparison are shown in Tables 1 and 2. These tables clearly state that `psf()` function is able to outperform the comparative time series prediction methods.

Additionally, the reader might want to refer to the results published in the original work [Martínez-Álvarez et al. \(2011a\)](#), in which it was shown that PSF outperformed many different methods when applied to electricity prices and demand forecasting.

## 5. Conclusions

This article is a thorough description of the PSF R package. This package is introduced to promote the algorithm Pattern Sequence based Forecasting (PSF) which was proposed by [Martínez-Álvarez et al. \(2008\)](#) and then modified and improved by [Martínez-Álvarez et al. \(2011a\)](#). The functions involved in the PSF package can be tested with time series data in any format like vector, time series, list, matrix or data frame. The aim of the PSF package is to simplify the calculations and to automate the steps involved in prediction while using PSF algorithm. Illustrative examples and comparisons to other methods are also provided.

Functions	<code>psf()</code>	<code>auto.arima()</code>	<code>ets()</code>
RMSE	22.11279	41.14366	52.29985

**Table 2:** Comparison of time series prediction methods with respect to RMSE values for sunspots dataset.



## Acknowledgements

We would like to thank our colleague, Mr. Sudhir K. Mishra, for his valuable comments and suggestions to improve this article. Thanks also goes to CRAN maintainers for needful comments on the submitted package. Finally, this study has been partially funded by the Spanish Ministry of Economy and Competitiveness and by the Junta de Andalucía under projects TIN2014-55894-C2-R and P12-TIC-1728, respectively.

## Bibliography

- G. Bontempi, S. B. Taieb, and Y. Le-Borgne. Machine learning strategies for time series forecasting. *Lecture Notes in Business Information Processing*, 138:62–77, 2013. [p2]
- J. Catalão, S. Mariano, V. Mendes, and L. Ferreira. Short-term electricity prices forecasting in a competitive market: a neural network approach. *Electric Power Systems Research*, 77(10):1297–1304, 2007. [p1]
- A. J. Conejo, M. A. Plazas, R. Espínola, and A. B. Molina. Day-ahead electricity price forecasting using the Wavelet transform and ARIMA models. *Power Systems, IEEE Transactions on*, 20(2):1035–1042, 2005. [p1]
- D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2:224–227, 1979. [p3]
- Y. Dodge. *The Oxford Dictionary of Statistical Terms*. The International Statistical Institute, 2003. [p2]
- J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974. [p3]
- Y. Fujimoto and Y. Hayashi. Pattern sequence-based energy demand forecast using photovoltaic energy records. In *Renewable Energy Research and Applications (ICRERA), 2012 International Conference on*, pages 1–6. IEEE, 2012. [p1]
- C. García-Martos, J. Rodríguez, and M. J. Sánchez. Mixed models for short-run forecasting of electricity prices: application for the spanish market. *Power Systems, IEEE Transactions on*, 22(2):544–552, 2007. [p1]
- C. H. Jin, G. Pok, H.-W. Park, and K. H. Ryu. Improved pattern sequence-based forecasting method for electricity load. *IEEE Transactions on Electrical and Electronic Engineering*, 9(6):670–674, 2014. [p1]
- L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2008. [p3]
- I. Koprinska, M. Rana, A. Troncoso, and F. Martínez-Álvarez. Combining pattern sequence similarity with neural networks for forecasting electricity demand time series. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 940–947. IEEE, 2013. [p1]
- M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, M. Studer, and P. Roudier. Package ‘cluster’, 2015. [p3]
- M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota. Modified pattern sequence-based forecasting for electric vehicle charging stations. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 710–715. IEEE, 2014. [p1]
- F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar-Ruiz. LBF: A labeled-based forecasting algorithm and its application to electricity price time series. In *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, pages 453–461. IEEE, 2008. [p1, 2, 3, 4, 8]
- F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar-Ruiz. Energy time series forecasting based on pattern sequence similarity. *Knowledge and Data Engineering, IEEE Transactions on*, 23(8):1230–1243, 2011a. [p1, 2, 3, 4, 8]
- F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar-Ruiz. Discovery of motifs to forecast outlier occurrence in time series. *Pattern Recognition Letters*, 32:1652–1665, 2011b. [p1]
- W. Shen, V. Babushkin, Z. Aung, and W. L. Woon. An ensemble model for day-ahead electricity demand time series forecasting. In *Future Energy Systems, 2013 ACM International Conference on*, pages 51–62, 2013. [p1]

A. Troncoso, J. M. R. Santos, A. G. Expósito, J. L. M. Ramos, and J. C. R. Santos. Electricity market price forecasting based on weighted nearest neighbors techniques. *Power Systems, IEEE Transactions on*, 22(3):1294–1301, 2007. [p1]

Neeraj Bokde  
Visvesvaraya National Institute of Technology, Nagpur  
North Ambazari Road, Nagpur  
India  
[neeraj.bokde@students.vnit.ac.in](mailto:neeraj.bokde@students.vnit.ac.in)

Gualberto Asencio-Cortés  
Universidad Pablo de Olavide  
ES-41013, Sevilla  
Spain  
[guaasecor@upo.es](mailto:guaasecor@upo.es)

Francisco Martínez-Álvarez  
Universidad Pablo de Olavide  
ES-41013, Sevilla  
Spain  
[fmaralv@upo.es](mailto:fmaralv@upo.es)

Kishore Kulat  
Visvesvaraya National Institute of Technology, Nagpur  
North Ambazari Road, Nagpur  
India  
[kdkulat@ece.vnit.ac.in](mailto:kdkulat@ece.vnit.ac.in)