PeerJ Preprints NOT PEER-REVIEWED

Demonstration of an Open Source Platform for Reproducible Comparison of Predictive Models

Neeraj Bokde[1], Kishore Kulat[1]
[1]Department of Electronics and Communication,
Visvesvaraya National Institution of Technology, Nagpur, India

Corresponding Author:
Neeraj Bokde
Department of Electronics and Communication,
Visvesvaraya National Institution of Technology, Nagpur, India
Email address: neerajdhanraj@gmail.com

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

Demonstration of an Open Source Platform for Reproducible Comparison of Predictive Models

Neeraj Bokde
(neerajdhanraj@gmail.com)

Abstract:
This paper discusses about a tool *PredictTestbench*, which is an R package which provides a testbench to do comparison of prediction methods. This package compares a proposed time series prediction method with other default methods like Autoregressive integrated moving average (ARIMA) and Pattern Sequence based Forecasting (PSF). The testbench is not limited to these methods. It allows user to add or remove multiple numbers of methods in the existing methods in the study. By default, testbench compares different imputation methods considering different error metrics RMSE, MAE or MAPE. Along with this, it facilitates user to add new error metrics as per requirements. The simplicity of the package usage and significant reduction in efforts and time consumption in state of art procedure, adds valuable advantage to it. The aim of the testbench is reduce the efforts for coding, experiments on output visualization and time for different steps involved in such study. This paper explains the use of all functions in *PredictTestbench* package with the demonstration of examples.

Literature Review :

Prediction techniques find usage in large numbers of applications. Since last two decades, developed countries have made tremendous change in traffic scenario with Intelligent Traffic System and Controls. All of this was become possible only because of accurate prediction techniques used to forecast the future traffic with higher accuracy. Economics, ecology, environmental studies, market research, medicine science, life science, physical science and many more research areas are using these predictive techniques for better improvements in controls and policy making ability. This section discuss about various articles focusing on predictive techniques proposed in recent past.

A new methodology to predict chaotic time series based on Recurrent Predictor Neural Network (RPNN) is proposed in article [1]. This method is studied long term predictions done with multistep predictions. In simulation, this methods was compared with Kalman filter and Universal learning network (ULN) method for 6 & 12 months ahead predictions. These methods were compared with two performance measures Root Mean Squared Errors (RMSE) and Prediction Accuracy (PA). Article [2] proposed a label based forecasting method Pattern Sequence based Forecasting (PSF) for energy consumption time series data. The novelty of method is the use of labels for forecasting instead of use of actual data. Author stated that PSF method outperforms methods like Naive, ANN, ARIMA, mixed models and WNN with respect to Mean error relative (MER) and Mean Absolute Errors (MAE) metric parameters. The study is done with best case and worst case one step ahead forecasting. Then drawbacks of PSF method like excessive time consumption is overcomed

PeerJ Preprints NOT PEER-REVIEWED

with introducing seasonal parameter in PSF as discussed in [19]. In article [3], Least Square Support Vector Machine (LSSVM) was used for financial time series prediction. In this study, LSSVM was compared with other predictive methods with one step ahead prediction performance. Similarly, in this article, prediction of energy consumption of electric vehicle is compared with Historic average, KNN, weighted KNN and lazy learning with performance metric Mean Absolute Percentage Errors (MAPE).

In [4], one day ahead electricity load prediction is done and compared with other competitive methods. There are large number of such applications of predictive analysis in areas like Renewable energy [5], soft computing [6], traffic volume forecasting [7]. The area of applications is not limited to these fields and almost all of such articles compared the proposed predictive method with other methods with performance metrics including RMSE, MAE, MAPE and MSE. Some of these articles compared predictive methods on basis of long term prediction errors whereas other articles used one step ahead forecasting method to calculate error with respect to original observed data. The time consumption for computation of the predictive method on particular hardware can also be one of the metric parameter for comparison used in [6].

In article [8], author stated that seasonal ARIMA model coupled with a Kalman filter is the most accurate model for short term prediction. This algorithm was compared with Random Walk, Seasonal Means, ARIMA with Kalman filter, SARIMA model with maximum likelihood fitting, SARIMA with Kalman filter, ANN, SVR with RBF kernel, SVR with RBF kernel multiplied by a seasonal kernel and SVR with linear seasonal kernel using MAPE as performance measures. Lippi M et. al. [8] observed that. Though many researchers are proposing many new predictive approaches, the results and predictions discussing about new algorithms employs different setting, dataset and performance measurements. This leads to make it difficult to understand advantages and disadvantages. Hence, a common platform with easy handling capacity is an demanding tool for such comparison of predictive models.

Introduction to *PredictTestbench* Package:

The procedures for prediction methods comparison are mostly application specific and most of the researchers use almost same procedure for it. The R package, *PredictTestbench* provides a handy platform for comparison of prediction methods. It functions like a testbench which allows user to create environment as per their requirements for comparison study. While evaluating comparison of proposed predictive method with other methods, researchers evaluate the performance of existing benchmarked methods and compare them with proposed method. Most of the times, these studies are performed manually step by step. This leads to more time consumption in analysis and may prone to errors in the study. Also, the lack of a common platform for comparative study can be one of the factor to failure or improper implementation of the study. Along with this, the reproducibility of a predictive model becomes limited due to lack of a reproducible platform for it. To eliminate these problems and difficulties in the state of art methods of prediction method comparative studies, the *PredictTestbench* package as a testbench is introduced. This testbench is broadly classified in two major methods for comparison. The former method is

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

long term prediction comparison whereas another method is one/two step ahead forecasting study. The syntax and possible input parameters for the functions included in these methods are discussed below.

Long term time series prediction comparison:

Long term time series prediction methods comparison is the comparison of different predictive methods with respect to values predicted by each method under study. In the testbench, two predictive methods are available by default. These methods include ARIMA [9] and PSF [2]. Analysis for ARIMA method is performed with *auto.arima()* function in forecast R Package [10]. This function implements the best suitable model in ARIMA including Seasonal ARIMA according to time series characteristics of the data under study. Whereas PSF algorithm is performed with R package, PSF [11][18]. Though these algorithms are compared by default, the testbench is not limited to these functions. It allows user to add multiple predictive methods in testbench. Also, testbench allows user to remove the unwanted method with very easy and user friendly steps. It is possible to compare different predictive methods with different error performance metrics including RMSE, MAE and MAPE. Along with this, the testbench allows user to append other error performance metrics as per requirements. Long term prediction is done with four functions *prediction_errors(), prediction_append(), prediction_remove()* and *plot_predictions()*.

This long term prediction in testbench is initiated with function *prediction_errors()* which takes various input parameters as discussed in *Table 1* with their default values. This function returns different parameters including type of error metric used for comparison, original time series data used as reference and the performance of each prediction method. This performance values for each predictive method include sub-parameters like predicted values for time series, the error values with respect to original observed data and the time consumed by the method to predict the future values. This output along with performance values is referred as prediction profile in this paper.

Along with this, addition of new predictive method can be done with function *prediction_append()*. The syntax and usage of this function is similar to *prediction_errors()* function, except *existing_method()* as an input parameter, which is one of the profile generated by *prediction_errors(), prediction_append()* or *prediction_remove()* functions as discussed in *Table 2*. This parameter is considered as base prediction profile over which extra method of prediction is to be added for comparison. Similarly, *prediction_remove()* function is used to omit the undesired prediction method in the study. This function takes prediction profile as input parameter along with the index of the method which is to be removed from testbench. Finally, any of these prediction profile generated by *prediction_errors(), prediction_append()* or *prediction_remove()* functions can be plotted with *plot_prediction()* function.

Parameter	Type	Description	Default Value
dataIn	Integer Vector	It should be a time series data in vector form of integers. It is considered as input data for prediction methods	Ramp shaped repeated Time

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

		comparison.	Series Data
nextVal	Integer	It is a number of values that are to be predicted by prediction methods.	*10*
errorParameter	Integer	It is an type of Error Metric which is to be studied in prediction method comparison. Different possible values for *errorParameter* are: '*1*' - RMSE '*2*' - MAE '*3*' - MAPE '*4*' - Add New Error Metric	'*1*' - RMSE
MethodPath	Character	It is the path of a function which contains code for prediction method which is to be added in testbench. It should be in Source format as "*source('~/abc/function.R')*".	NULL
MehodName	Character	It is the title given for newly proposed method with *MethodPath* parameter.	"*Proposed_Method*"

Table 1 List of Input Parameters for '*prediction_errors()*' function discussing about Type and its corresponding default values.

One/Two Step Ahead forecasting:

In many research articles, the comparison of prediction algorithms is done with one or two step ahead forecasting methods. In one step ahead forecasting, the time series data is partitioned into training and test dataset. The procedure starts with prediction of one step ahead prediction with respect to training data. Then the error (RMSE, MAE, MAPE or any other) between the predicted value and test data at that point is calculated. This step by step procedure is followed till all equivalent predicted data for test data is obtained. In similar fashion, two step ahead prediction is done by predicting second step ahead value and following the same procedure as one step ahead forecasting.

In the testbench, one/two step ahead forecasting is done with *step_ahead_forecast()* function. This function takes various input parameters including time series data. The detailed description and its default values for these parameters are discussed in *Table 3*. This function returns a plot with original data and its respective predicted data obtained by a particular predictive method with addition of positive and negative ranges of standard deviation. It also returns the error values between predicted and original test data, as per *errorParameter* value selected by user. Next section discuss about the working of the testbench with few examples.

Parameter	Type	Description	Default Value
existing_method	List	It is a list in format returned by *prediction_errors()*, *prediction_append()* or *prediction_remove()* functions. This list is also referred as Prediction Profile in this paper.	NULL

Table 2 List of additional Input Parameter for '*prediction_append()*' function discussing about Type and its corresponding default values

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

Parameter	Type	Description	Default Value
dataIn	Integer Vector	It should be a time series data in vector form of integers. It is considered as input data for prediction methods comparison.	Ramp shaped repeated Time Series Data.
trainedData	Integer	It is a partition point of input data '*dataIn*'. Data till this point is referred as Training data and rest of data is used for Testing.	*75%* data as Training dataset and rest *25%* as Testing dataset.
errorParameter	Integer	It is an type of Error Metric which is to be studied in prediction method comparison. Different possible values for *errorParameter* are: '*1*' - RMSE '*2*' - MAE '*3*' - MAPE '*4*' - Add New Error Metric	'*1*' - RMSE
MethodPath	Character	It is the path of a function which contains code for prediction method which is to be added in testbench. It should be in Source format as "*source('~/xyz/function.R')*".	NULL
stepSize	Integer	It is indication of size for step ahead forecasting. Possible values are: '*1*' - One step ahead forecasting '*2*' - Two step ahead forecasting	'*1*' - One step ahead forecasting

Table 3 List of Input Parameters for '*step_ahead_forecast()*' function discussing about Type and its corresponding default values

Demonstration of *PredictTestbench* Package with Examples:

This example uses the Santa Fe Laser dataset [12] to show the comparison for long term prediction outcome and performance of one/two step ahead prediction. The Santa Fe Laser dataset contains a chaotic laser time series obtained in a physics laboratory experiment as shown in *Figure 1*. In this dataset, univariate time series data is involved with one variable. For present study, 1000 observations in dataset is used. Out of this, first 800 values are used for training and rest are used for testing.

PeerJ Preprints NOT PEER-REVIEWED

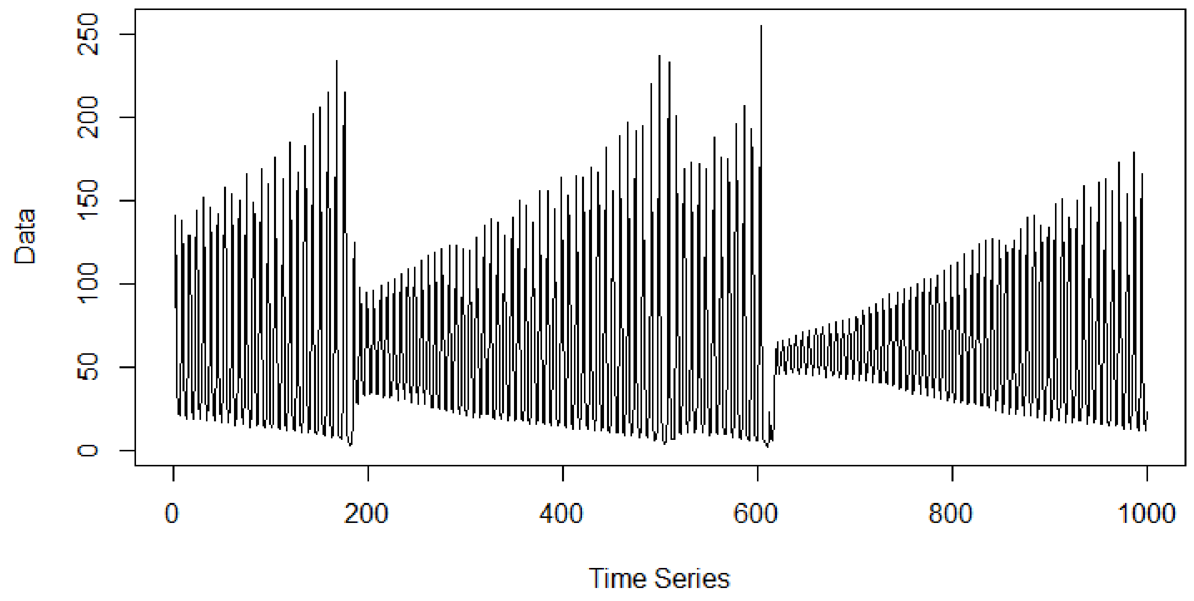

Figure 1 The Santa Fe Laser dataset containing chaotic laser time series

The long term time series prediction is done with various functions as discussed in earlier section. This method starts with *prediction_errors()* function which takes '*dataIn*' as input time series data and '*errorParameter*' as the selection of error metric for comparison of different predictive methods. As discussed in *Table 1*, the default value for *errorParameter* is *1*, which decides RMSE as a error performance metric. At the simplest form, *prediction_errors()* performs as shown below.

```
library(PredictTestbench)
a <-prediction_errors(dataIn = aa, nextVal = 5)
a
plot_predictions(a)
```

This function returns the prediction profile as shown below. This profile consists of predicted values, its RMSE value corresponding to ARIMA and PSF algorithms. It also returns the plot for predicted values as shown in *Figure 2*.

```
## $Parameter
## [1] "RMSE Plot"
##
## $Desired_Prediction
##  V1996  V1997  V1998  V1999 V11000
##     61     20     12     13     23
##
```

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

```
## $ARIMA_Method_Prediction
## [1]  84.975031  11.532186   1.027564  12.892061  32.423839  82.915630
## [7] 135.496897 126.033239  58.639438   2.794173
##
## $ARIMA_Method_Error
## [1] 54.92755
##
## $ARIMA_Execution_Time_in_Seconds
## [1] 4.715345
##
## $PSF_Method_Prediction
## [1] 36.80054 36.80054 36.80054 36.80054 36.80054
##
## $PSF_Method_Error
## [1] 21.16535
##
## $PSF_Execution_Time_in_Seconds
## [1] 26.42523
```

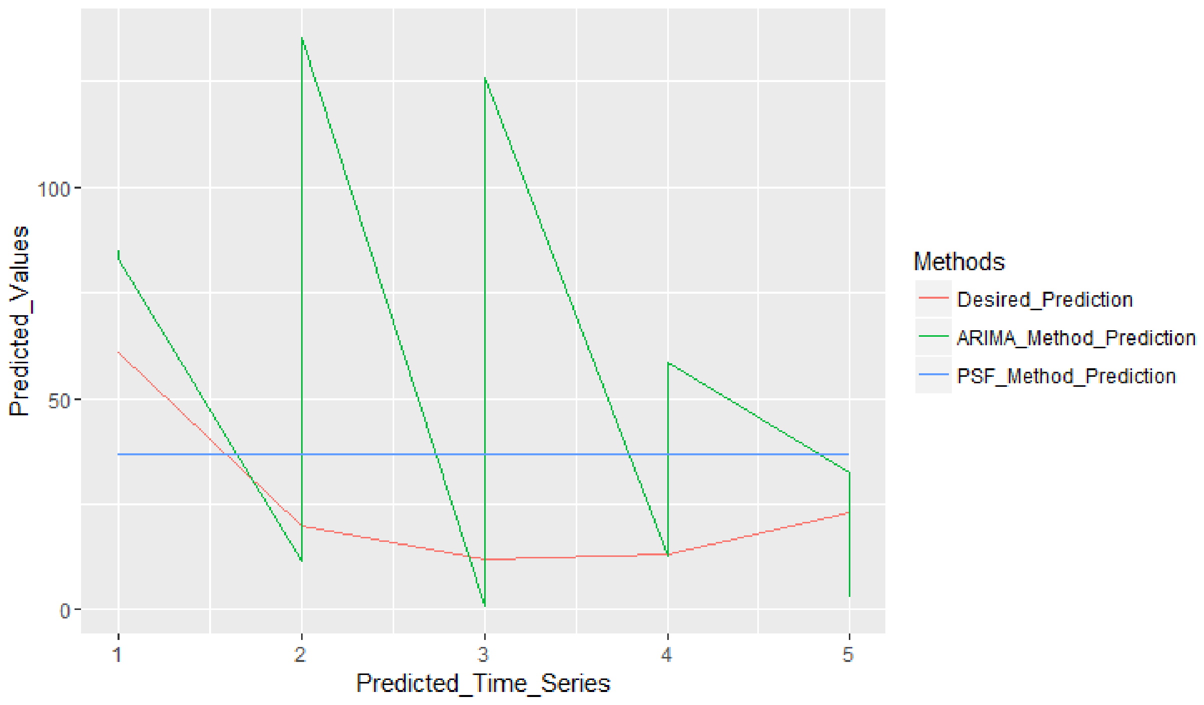

Figure 2 Plot of predicted values obtained with '*prediction_errors()*' function for variable '*a*'

Also, this function allows user to add one more predictive method for comparison in the study. This is done with *MethodPath* and *MethodName* parameters. *MethodPath* is the path for the function file used for the new prediction algorithm to be added in testbench. Let us consider the following code which returns the predictive values with a random algorithm under title of '*Proposed_Method*'. While constructing such additive functions in testbench, few precautions are to be taken under consideration such that the additive functions should take time series data as input and should return the predicted values. One sample example for such function is shown below.

PeerJ Preprints NOT PEER-REVIEWED

```
====================================================================
SPSF <- function(dataIn, seas){
  library(PSF)
  a <- AUTO_PSF(data_in = dataIn, next_val = 1)$Predicted_Values
  return(a)}
====================================================================
```

The path of this additive function should be captured in Source format as `"source('~/SeasonalPSF/R/SPSF.R')"`. This path of the function can be obtained easily with 'Source' menu in RStudio. In addition to this, the nomenclature of the newly added prediction method is done with *MethodName* parameter. The following code for *prediction_errors()* shows the procedure to add an additive method for prediction in the testbench.

```
b <-  prediction_errors(dataIn = aa,
       nextVal = 5,
       MethodPath = "source('~/SeasonalPSF/R/SPSF.R')",
       MethodName = "Proposed_Method")
b
plot_predictions(b)
```

The output generated with this function is similar to that of earlier function except the addition of prediction profile of newly added method in testbench. These prediction profile obtained in these method can be plotted with *plot_prediction()* function as shown in *Figure 3*.

```
## $Parameter
## [1] "RMSE Plot"
##
## $Desired_Prediction
##  V1996  V1997  V1998  V1999 V11000
##     61     20     12     13     23
##
## $ARIMA_Method_Prediction
## [1]  84.975031  11.532186   1.027564  12.892061  32.423839  82.915630
## [7] 135.496897 126.033239  58.639438   2.794173
##
## $ARIMA_Method_Error
## [1] 54.92755
##
## $ARIMA_Execution_Time_in_Seconds
## [1] 4.171517
##
## $PSF_Method_Prediction
## [1] 36.80054 36.80054 36.80054 36.80054 36.80054
##
## $PSF_Method_Error
## [1] 21.16535
##
## $PSF_Execution_Time_in_Seconds
## [1] 29.2653
##
## $Proposed_Method_Prediction
```

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

```
## [1] 37.46597  37.46597 100.88991 100.88991 100.88991  37.46597  37.46597
## [8] 37.46597  37.46597  37.46597
##
## $Proposed_Method_Error
## [1] 49.8669
##
## $Proposed_Method_Execution_Time_in_Seconds
## [1] 36.46742
```

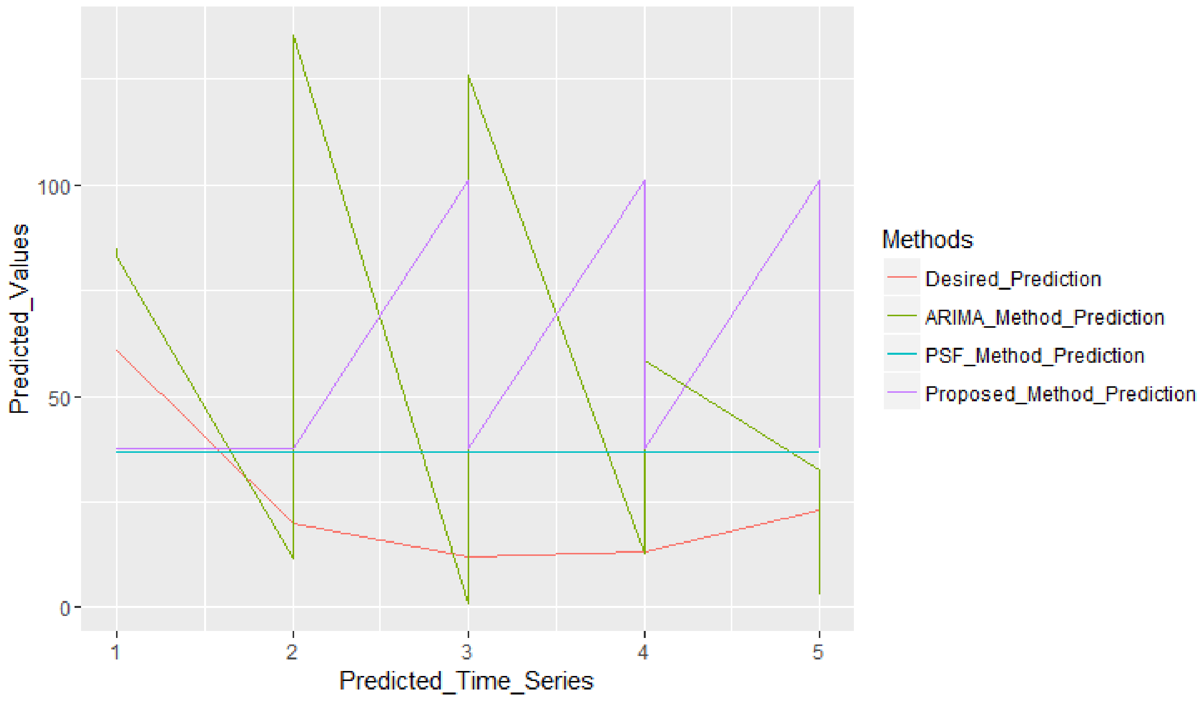

Figure 3 Plot of predicted values obtained with '*prediction_errors()*' function for variable '*b*'

The testbench allows extension of *prediction_errors()* function, as well. Testbench makes it possible through *prediction_append()* function which allows user to add more and more additive prediction methods in comparison study. As discussed earlier, this function, input parameters and its syntax are similar to *prediction_errors()* function. The *prediction_append()* function considers list '*b*' as input error profile and compares all those prediction methods in list '*b*' with another method seasonal ARIMA introduced in following code.

```
c <-   prediction_append(existing_method = b,
       dataIn = aa,
       nextVal = 5,
       MethodPath = "source('~/testing2/SARIMA.R')",
       MethodName = "Seasonal_ARIMA")

c

plot_predictions(c)
```

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

The output obtained with this function is shown in list form as follows. It contains predicted values, prediction errors and corresponding time consumption for the calculation for Seasonal ARIMA method in addition to data already available in list '*b*' . It also plot the predicted values obtained in methods attached with the testbench as shown in *Figure 4*.

```
## $Parameter
## [1] "RMSE Plot"
##
## $Desired_Prediction
##  V1996  V1997  V1998  V1999 V11000
##     61     20     12     13     23
##
## $ARIMA_Method_Prediction
## [1]  84.975031  11.532186   1.027564  12.892061  32.423839  82.915630
## [7] 135.496897 126.033239  58.639438   2.794173
##
## $ARIMA_Method_Error
## [1] 54.92755
##
## $ARIMA_Execution_Time_in_Seconds
## [1] 4.171517
##
## $PSF_Method_Prediction
## [1] 36.80054 36.80054 36.80054 36.80054 36.80054
##
## $PSF_Method_Error
## [1] 21.16535
##
## $PSF_Execution_Time_in_Seconds
## [1] 29.2653
##
## $Proposed_Method_Prediction
## [1] 37.46597  37.46597 100.88991 100.88991 100.88991  37.46597  37.46597
## [8] 37.46597  37.46597  37.46597
##
## $Proposed_Method_Error
## [1] 49.8669
##
## $Proposed_Method_Execution_Time_in_Seconds
## [1] 36.46742
##
## $`Seasonal ARIMA_Prediction`
## [1] 37.06380 53.69645 58.18571 59.39739 59.72443 59.81270 59.83653
## [8] 59.84296 59.84470 59.84516
##
## $`Seasonal ARIMA_Prediction_Error`
## [1] 38.43464
##
## $`Seasonal ARIMA_Prediction_Execution_Time_in_Seconds`
## [1] 0.1413751
```

For first three values of *errorParameter*, fixed error metrics are assigned as shown in *Table 1* and default value of this parameter is *1*. With the selection of *errorParameter* as *4* , it allows

PeerJ Preprints NOT PEER-REVIEWED

user to select a new error metric other than already available in testbench. The syntax of this parameter is as shown below.

```
errorParameter = c(4, “source('~/abc/function.R')”, “label_name”)
```

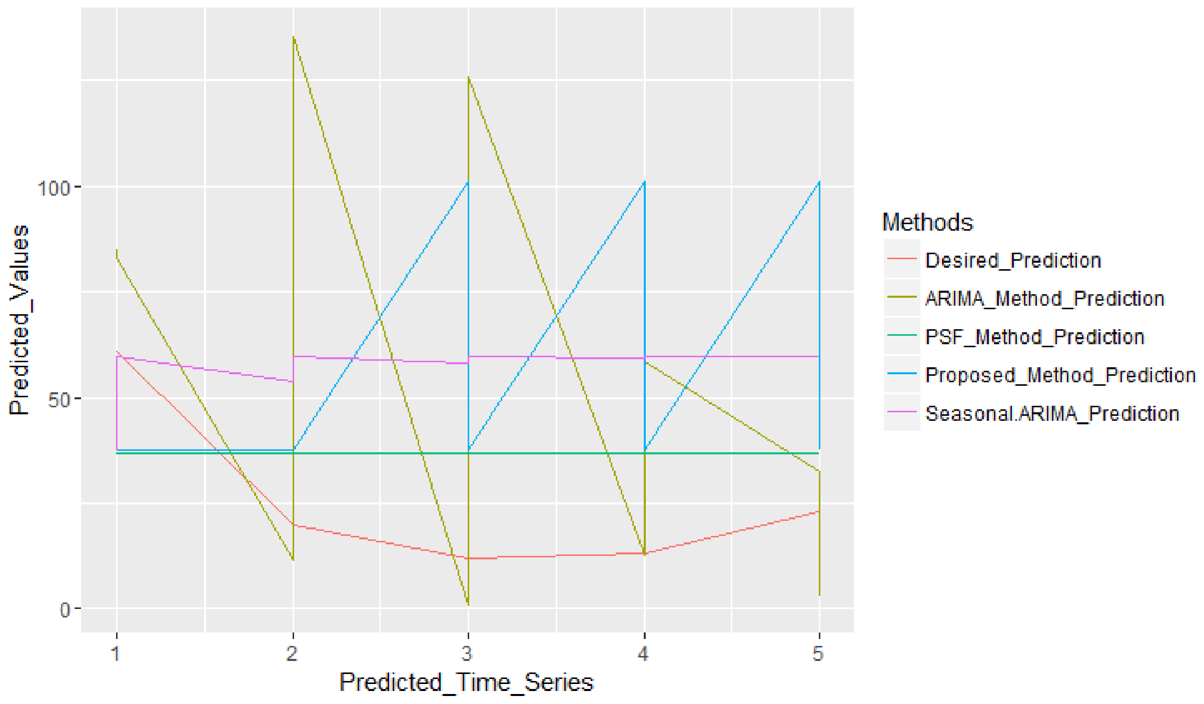

Figure 4 Plot of predicted values obtained with '*prediction_append()*' function for variable '*c*'

where, *4* is the mandatory integer to indicate that a new error metric is to be attached. The location of code for new error metric and its corresponding label are provided with second and third terms. The code for new error metric should be in function form which is to be written in an R script. This function should contain two input parameters as original and its equivalent predicted data series and should return the error corresponding to the error function.

Finally, the only left out function in long term time series data prediction function *prediction_remove()* manages to remove the unwanted method in testbench as explained with following code.

```
d <-   prediction_remove(existing_method = c, index_number = 3)
plot_predictions(d)
```

This function removes the prediction method labeled as '*Proposed_Method*' which is having index number *2* in prediction profile '*c*' and corresponding new list '*d*' along with a plot is get generated as shown in *Figure 5*. In this way, these functions allowed user to add desired and remove unwanted prediction methods as per requirement.

Another feature of the testbench is one/two step ahead forecasting technique for comparison with *step_ahead_forecast()* function. This function generates a plot and calculate

PeerJ Preprints NOT PEER-REVIEWED

prediction error for one/two step ahead forecast of a predictive algorithm. Consider following code which takes time series data under study and performs one/two step ahead forecast.

This function consumes various input parameters as discussed in *Table 3*. In present study, *trainedData* parameter is kept 800 which partitioned the whole data at time index of 800 in two time series. The former time series data will act as training dataset and rest of time series data acts as test dataset. With selection of *errorParameter* as *1*, this function returns the RMS error calculation based one step ahead prediction of ARIMA method.

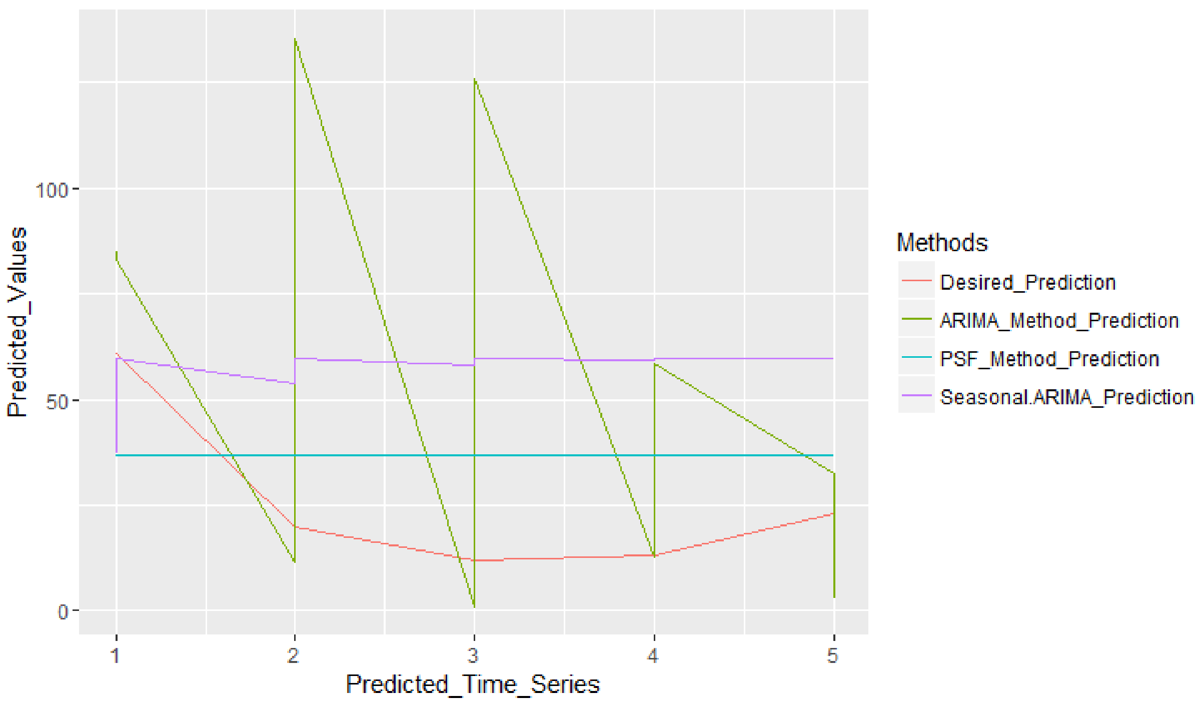

Figure 5 Plot of predicted values obtained with '*prediction_remove()*' function for variable '*d*'

```
e <-   step_ahead_forecast(dataIn = aa, trainedData = 800,
       MethodPath = "source('~/testing2/arima.R')",
       errorParameter = 1,
       stepSize = 1)

e

## $Plot
```

PeerJ Preprints NOT PEER-REVIEWED

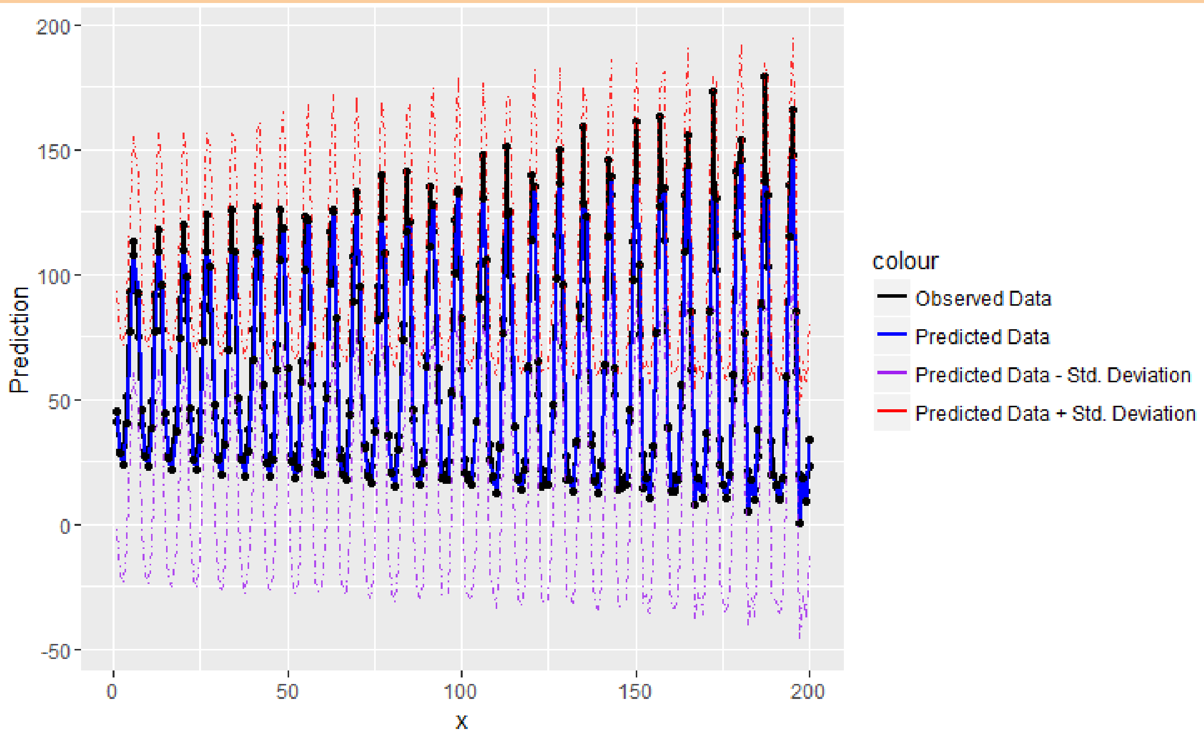

Figure 6 One step ahead forecasting for ARIMA method with '*step_ahead_forecast()*' function

```
##
## $`RMSE Value`
## [1] 13.22782
```

Similarly, two step ahead forecasting for same dataset with similar partition is done with following code.

```
f <-    step_ahead_forecast(dataIn = aa,
        trainedData = 800,
        MethodPath = "source('~/testing2/arima.R')",
        errorParameter = 1,
        stepSize = 2)
f

## $Plot
```

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

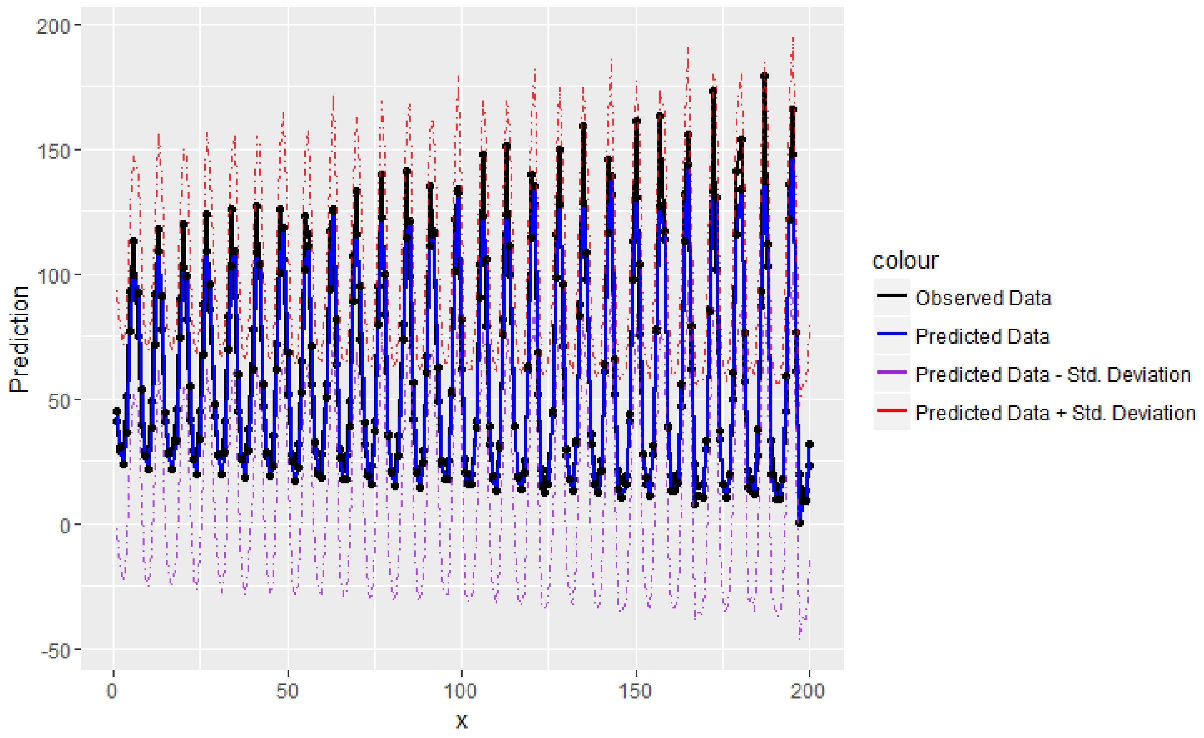

Figure 7 Two step ahead forecasting for ARIMA method with '*step_ahead_forecast()*' function

```
##
## $`RMSE Value`
## [1] 13.26184
```

The observable change in one step and two step ahead prediction errors for ARIMA prediction algorithm can be noticed in last two code outcomes.

Conclusion:

This paper discussed about *PredictTestbench*, an R package [13] which works as a testbench to compare time series prediction methods. The usability of this package is demonstrated with an example. By default, this testbench compares a proposed method by user with existing prediction methods (ARIMA and PSF) with RMSE, MAE or MAPE as error metrics. Also, the testbench allows to add new multiple imputation methods in comparison along with default methods. This package may support the prediction methods compiled with C, Fortran, C++, Java, Python or Matlab languages with the help of R packages like Rcpp [14], rJava [15], rPython [16] and matlabr [17]. Also, the example discussed about the procedure to add new error metric in addition to existing error metrics. The simple working of testbench to add and remove multiple prediction methods, makes it more robust and useful. The results discussed in this paper are performed using R 3.2.5. The *PredictTestbench* package is available at CRAN repository with URL https://cran.r-project.org/web/packages/PredictTestbench/index.html.

http://www.neerajbokde.com/

PeerJ Preprints NOT PEER-REVIEWED

References:

[1] Han M, Xi J, Xu S, Yin FL (2004) Prediction of chaotic time series based on the recurrent predictor neural network. Signal Processing, IEEE Transactions on.;52(12):3409-16.

[2] Martinez Alvarez F, Troncoso A, Riquelme JC, Aguilar Ruiz JS (2011) Energy time series forecasting based on pattern sequence similarity. Knowledge and Data Engineering, IEEE Transactions on.;23(8):1230-43.

[3] Van Gestel T, Suykens JA, Baestaens DE, Lambrechts A, Lanckriet G, Vandaele B, De Moor B, Vandewalle J (2001) Financial time series prediction using least squares support vector machines within the evidence framework. Neural Networks, IEEE Transactions on.;12(4):809-21.

[4] Clements AE, Hurn AS, Li Z (2014) Forecasting day-ahead electricity load using a multiple equation time series approach. European Journal of Operational Research.

[5] Doucoure B, Agbossou K, Cardenas A (2016) Time series prediction using artificial wavelet neural network and multi-resolution analysis: Application to wind speed data. Renewable Energy.;92:202-11.

[6] Peng HW, Wu SF, Wei CC, Lee SJ (2015) Time series forecasting with a neuro-fuzzy modeling scheme. Applied Soft Computing;32:481-93.

[7] Xie Y, Zhao K, Sun Y, Chen D (2010) Gaussian processes for short-term traffic volume forecasting. Transportation Research Record: Journal of the Transportation Research Board. (2165):69-78.

[8] Lippi M, Bertini M, Frasconi P (2013) Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. Intelligent Transportation Systems, IEEE Transactions on;14(2):871-82.

[9] Hillmer SC, Tiao GC (1982) An ARIMA-model-based approach to seasonal adjustment. Journal of the American Statistical Association;77(377):63-70.

[10] Hyndman RJ, Khandakar Y (2007) Automatic time series for forecasting: the forecast package for R. Monash University, Department of Econometrics and Business Statistics.

[11] Bokde Neeraj (2016) PSF: Algorithm for Pattern Sequence Based Forecasting. R package version 0.1.1. https://CRAN.R-project.org/package=PSF

[12] Santa Fe Laser time series. http://www-psych.stanford.edu/andreas/Time-Series/SantaFe.html

[13] Bokde Neeraj (2016). PredictTestbench: Test Bench for Comparison of Data Prediction Models. R package version 1.1.1 https://cran.r-project.org/package=imputeTestbench

[14] Eddelbuettel D, François R, Allaire J, Chambers J, Bates D, Ushey K (2010) Rcpp: Seamless R and C++ integration. Journal of Statistical Software;40(8):1-8.

[15] Urbanek s (2016) rJava: Low-Level R to Java Interface, 2016. URL https://CRAN.R-project.org/package= rJava. R package version 0.9-8

[16] Ancona D, Ancona M, Cuni A, Matsakis ND (2007) RPython: a step towards reconciling dynamically and statically typed OO languages. InProceedings of the 2007 symposium on Dynamic languages (pp. 53-64). ACM.

[17] J. Muschelli (2015) matlabr: An Interface for MATLAB using System Calls. URL https://CRAN.Rproject.org/package=matlabr. R package version 1.1.

[18] Bokde, N and Kulat, K (2016) PSF : Introduction to R Package for Pattern Sequence Based Forecasting Algorithm. *arXiv preprint arXiv:1606.05492*

PeerJ Preprints NOT PEER-REVIEWED

[19] Bokde N, Gupta A, Kulat K. (2016) Introduction of seasonality concept in PSF algorithm to improve univariate time series predictions. *PeerJ Preprints* 4:e2184v1 https://doi.org/10.7287/peerj.preprints.2184v1