INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 2)
Available online at: www.ijariit.com

Time Series Forecasting using R

Pankaj Garg
pankajgarg@mait.ac.in
Maharaja Agrasen Institute of Technology, Rohini, Delhi

Harshit Arora
harshit.arora26@icloud.com
Maharaja Agrasen Institute of Technology, Rohini, Delhi

Rishabh Gambhir
rishabh.gambhir96@yahoo.com
Maharaja Agrasen Institute of Technology, Rohini, Delhi

ABSTRACT

R provides many features to analyze and visualize time-series data. Time series modeling and forecasting are used for various business applications, such as exploring sales patterns and trends, predicting future sales and detecting unusual phenomena. This allows companies to reduce risk, increase profit and achieve cost-effectiveness. Several machine learning algorithms exist that can be used for forecasting, some of which already exist in the forecast package in R. In this paper, we aim to explore some of the commonly used techniques for predictions.

Keywords: *Machine Learning, Time Series, R, Forecast, ARIMA.*

1. INTRODUCTION

Forecasting refers to the process of making predictions. Observations that are recorded at regular time intervals form a time-series. For example, weekly sales history, daily stock forecast, monthly crop production are some of the common time-series objects. Forecasts can be needed only hours in advance, or even months in advance. Time Series forecasting is achieved by converting the pre-existing data into a time series object, data cleaning and manipulation, followed by analysis of the data to create an efficient model which can provide the adequate forecasts. Often, data is visualized by the use of graphs. Graphs enable the characteristics of the data to be explored, including patterns, trends, and unusual observations.
This paper describes the process of data exploration, common forecasting models used, evaluating the accuracy of models in R.

2. UNDERSTANDING TIME SERIES

Once data is imported into R (most commonly from Excel, SPSS, Stata or TXT files), we can convert it into a time series object, commonly referred by R as ts. The ts() function is used to convert a vector into an R time series object, the formula for which is ts(vector, start=, end=, frequency=). Here, start and end are the times of the first and last observation and frequency is the number of observations in a unit time (For example, frequency=1 will mean annual observations, whereas frequency=52 will mean weekly observations)

Once the time series object is available, we can plot it by using the plot.ts() function in R. Plotting the time series allows us to understand how the data changes over time and helps in determining patterns.

Time series can be additive or multiplicative. Whether a time series is additive or multiplicative is described by the components, which are explored by decomposing the time series. The three components are:

- trend component, which describes how the changes occur overall, i.e., whether the trend is upward or downward
- seasonality component, which describes how things change within a given period (within a week, month, or year)
- Residual/error/irregular component, which cannot be explained by the trend. This component is randomness which is also known as white noise.

For Additive Time Series,
$\boldsymbol{Y_t = S_t + T_t + E_t}$

For Multiplicative Time Series,
$\boldsymbol{Y_t = St_t \times T_t \times E_t}$

where $\boldsymbol{Y_t}$ = Data
$\mathbf{T_t}$ = Trend component
$\boldsymbol{E_t}$ = Error or white noise

It is often preferred to convert a multiplicative time series into an additive time series, which can be generally done by taking the log of the multiplicative time series.

2.1 Stationary Time Series

A stationary time series has statistical properties (mean, autocorrelation, and variance) that do not change over time. Many forecasting models rely on the assumption that the time series can be made approximately stationary, or "stationarized" using mathematical transformations. A stationarized series is preferred because it is easier to predict because we can assume that it's statistical properties will not change over time.

A time series is said to be stationary if:

- The mean value of time-series does not change over time, i.e. , the trend component is nullified.
- The variance does not increase over time.
- Seasonality is minimal.

2.2 Decomposing the Time Series

For time-series, decomposition refers to the process of splitting the time-series into its components. The "decompose ()" or "stl()" function is used to split the time series into its three components.
The "decompose()" function returns a list object in which the estimates of the seasonal component, trend component and irregular component are stored as named elements, known as "seasonal", "trend", and "random".

For time-series that follow the additive model, seasonal adjustment is done by estimating the seasonal component and subtracting this component from the initial time series. The seasonally adjusted time series can be visualized using the plot() function. This time series does not contain the seasonal variation- instead contains only the trend and residual component.

3. METHODOLOGY FOR TIME-SERIES FORECASTING

- Data collection and cleaning
- Fitting into time series object
- Data decomposition and type identification
- Training and validating the model
- Using a benchmark model
- Improving accuracy using better fit model

4. CROSS-VALIDATION USING TRAINING AND TEST SETS

The easiest method to split the data for cross-validation is a division into two sets: a training set and a test set (or a validation set). Cross-validation is crucial because it provides an accurate assessment of the models used for forecasting. Any model is usually trained on the training test and then tested against the validation set. A common approach is the 80/20 rule. i.e, using 80% of the data for training, and 20% for testing. The performance of the model is then judged by the RMSE, or Root Mean Square Error.

4.1 Benchmark Models

Some models used for forecasting are simple to implement and understand, yet surprisingly effective. Benchmarking allows comparisons to occur by comparing the results of advanced models such as ARIMA against simple benchmarking models such as the Naive and Average model. In order for an advanced model to be effective, it must beat the accuracy of the benchmark models.

4.1.1 The Average Method

For the average method, as the name suggests, the forecasts of all predicted values are equal to the average of the past observations.

$$Y_{T+h} = (Y_1+Y_2...Y_T)/T$$

where h=1,2,3...

4.1.2 The Naive Method

The naive method or model is exclusive to time-series analysis. All predicted values, or forecasts, are set to be equal to the last known observation. Hence, all forecasts values will be Yt, where Yt is the last observed value.

$$Y_{T+h=} Yt$$

4.1.3 The Seasonal Naive Method

The seasonal naive model is a more advanced version of the naive model used for seasonal data, where each predicted value is set to be equal to the observed value from the previous season. For instance, A given value for the date 10th January 2017 will be set as the value from 10th January 2016.

4.2 Model Errors

The accuracy of a model is usually judged by the Root Mean Square Error (RMSE), also known as Root Mean Square Deviation (RMSD). It is a measure of the difference between predicted observations and the actual values. The individual differences are called residuals.

$$\mathrm{RMSE}_{fo} = [\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N]^{1/2}$$

Where:

- Σ = Summation
- Z_{fi} - Z_{oi} = Distance between each observation and the mean
- N = Sample size

4.3 The ARIMA Model

An autoregressive integrated moving average (ARIMA) model is fit to time-series data in order to get a better understanding of the data and to forecast the future values.

The equation for a stationary time series is a linear equation in which the predictors consist of lags of the dependent variable. ARIMA models are usually estimated by using the Box-Jenkins approach.

A nonseasonal ARIMA model is better known as an ARIMA(p,d,q) model, where:

p is the number of autoregressive terms
d is the number of non-seasonal differences required to achieve stationarity
q is the number of lagged forecast errors in the prediction equation

5. REFERENCES

[1] Juan Trujillo. 2011. A review of time series data mining - Engineering Applications of Artificial Intelligence, 24 (2011) 164–181. ELSEVIER.
[2] http://en.wikipedia.org/wiki/R(programming_language)
[3] Hyndman R.J., Khandakar Y. (2008): Automatic Time Series Forecasting: The forecast Package for R, Monash University, Journal of Statistical Software, Volume 27, Issue 3. (http://www.jstatsoft.org)
[4] Hyndman RJ, Koehler AB, Snyder RD, Grose S (2002). "A State Space Framework for Automatic Forecasting Using Exponential Smoothing Methods." International Journal of Forecasting, 18(3), 439–454.

