Time-Series Analysis and Forecasting of Foreign Exchange Rate of INR-USD Currency

submitted by

E B Benson
Cochin University of Science And Technology
Kerala

under the guidance of

R Team

FOSSEE, IIT Bombay

July 2024

Contents

1 Abstract

This study analyzes and forecasts daily INR-USD exchange rates from January 2000 to June 2024 using various time series models. Progressing from linear techniques like simple regression and ARIMA to nonlinear approaches such as Threshold Autoregressive (TAR) and Smooth Transition Autoregressive (STAR) models, the research aims to capture the complex dynamics of foreign exchange markets. Through rigorous evaluation using diagnostic tools, including ACF and PACF plots, residual analysis, and AIC, the study demonstrates that nonlinear models, particularly STAR, provide superior fit and forecasting capability. A seven-day forecast generated using the STAR model offers insights into short-term exchange rate trends, highlighting its effectiveness in capturing historical patterns and providing reliable forecasts. This research contributes to financial time series analysis by emphasizing the importance of nonlinear modeling in understanding and predicting forex market dynamics, with practical implications for traders, policymakers, and researchers.

2 Introduction

A wide range of tradings are happening in the currency exchange platform field. The INR-USD exchange is one of the most occurring exchanges in the world. So, it is crucial to predict the exchange rate equally for traders and analysts. There are a lot of time series models that can be used for the analysis and prediction of foreign exchange rates. This study investigates the behavior of daily exchange rates of the Indian rupee (INR) against the US dollar (USD) using time series analysis. The exchange rate between the Indian Rupee (INR) and the United States Dollar (USD) is a crucial indicator of economic health and international trade dynamics between India and the United States. The INR-USD exchange rate represents the value of the Indian Rupee against the United States Dollar. It determines the cost of exchanging Indian Rupees for U.S. Dollars and vice versa, impacting trade, investment, and economic policies. The INR-USD exchange rate fluctuations can significantly affect India's economy by influencing import and export costs, inflation rates, and foreign investment flows. This analysis delves into the daily noon exchange rates of INR against USD, with data from the Federal Reserve Bank New York, accessed via the FRED platform provided by the Federal Reserve Bank of St. Louis. The dataset covers the period from January 2000 to June 2024, encompassing critical economic events and shifts that have shaped the bilateral trade and investment landscape over the past decade. In this case study, we use different linear and nonlinear models such as linear, Auto-Regressive, ARIMA, and STAR.

3 Methodology

3.1 Data Collection

The INR-USD Exchange Rate dataset, which consists of the daily exchange rate of USD-INR, is used for the analysis. The data are noon buying rates in New York. The data starts from January 3, 2000, to June 14, 2024. The data is recorded daily at 4 PM DST and is provided with a 7-day frequency.

3.2 Data Sources

The data is sourced with the following descriptions:

- **Name:** H.10 Weekly Release [1]
- **Provider:** Federal Reserve Bank of New York
- **Currency:** INR-USD exchange rate
- **Frequency:** Daily
- **Time Period:** January 2000 to June 2024

The dataset consisted of USD exchange rates with various countries, but we focused on only the INR-USD exchange rate.

3.3 Data Description

The dataset is stored in the CSV format and has the dimension of 6381 rows and two columns. The description of headers/column names of the constructed dataset is given in the following table:

Table 1: Description of each header of the constructed dataset.

Attributes	**Description**	**Data Type**
Unique.Identifier..	The date of the recorded exchange rate.	character
H10.H10.RXI_N.B.IN	The exchange rate of the day at noon.	character

A random subset of size five is selected from the constructed dataset and shown below for clarity.

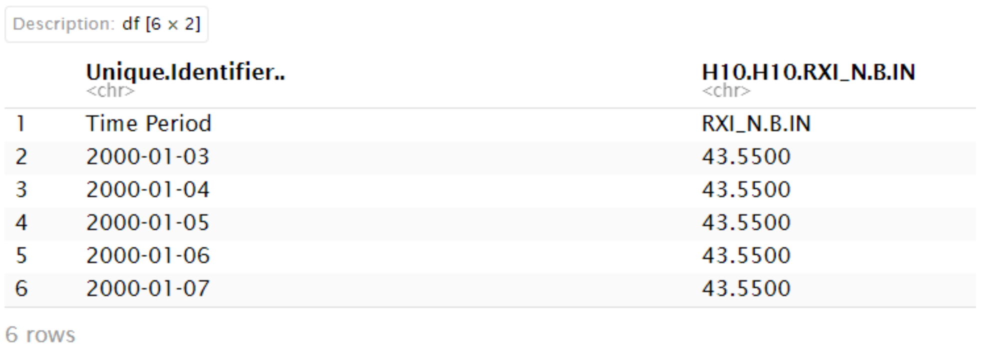
Table 2: A random subset of the dataset.

Unique.Identifier..	H10.H10.RXI_N.B.IN
2000-01-03	43.5500
2000-02-29	43.6500
2003-12-29	45.6600
2004-02-10	45.2200
2004-02-17	45.2800

3.4 Data Preprocessing

Let us import the data and read the head of the data along with a summary of the data to see how the data is distributed.

```
df <- read.csv("FRB_H10.csv")
head(df)
summary(df)
```



```
Description: df [6 x 2]

   Unique.Identifier..                        H10.H10.RXI_N.B.IN
   <chr>                                      <chr>
1  Time Period                                RXI_N.B.IN
2  2000-01-03                                 43.5500
3  2000-01-04                                 43.5500
4  2000-01-05                                 43.5500
5  2000-01-06                                 43.5500
6  2000-01-07                                 43.5500

6 rows
```

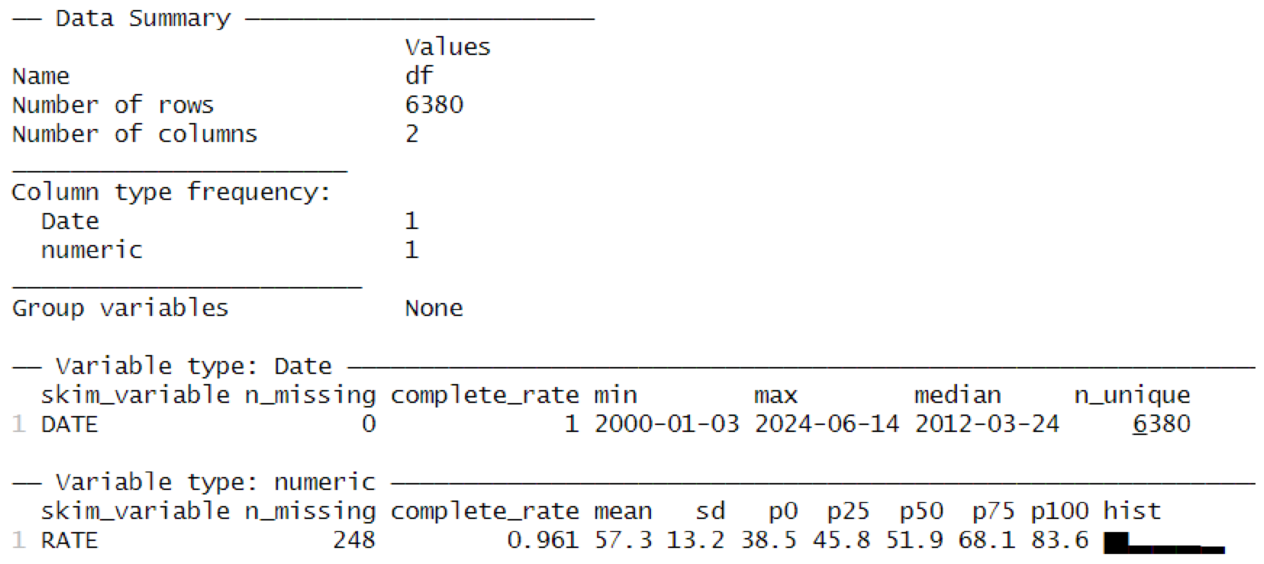
Figure 1: Exchange Rates of INR against USD

```
Unique.Identifier..  H10.H10.RXI_N.B.IN
Length:6381          Length:6381
Class :character     Class :character
Mode  :character     Mode  :character
```

Figure 2: Summary of unprocessed data

As we can see, there is an extra row stating the contents of the row, and then only the original data starts. So we shall be removing the first row. In the summary, we can see that both the data values are in characters datatypes and not in their respective datatypes, so we shall also make them into the date and numeric format. We also convert the column names to DATE and RATE for easiness of calling them. These are done using the following commands along with a summary of the data after preprocessing using the **skim(x)** function in the skimr package of R.

```r
df <- df[-1,]
rownames(df) <- NULL
colnames(df) <- c("DATE", "RATE")
df$DATE <- as.Date(df$DATE)
df$RATE <- as.numeric(df$RATE)
skim(df)
```



```
── Data Summary ────────────────────────
                           Values
Name                       df
Number of rows             6380
Number of columns          2
_______________________
Column type frequency:
  Date                     1
  numeric                  1
________________________
Group variables            None

── Variable type: Date ──────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate min        max        median     n_unique
1 DATE                  0             1 2000-01-03 2024-06-14 2012-03-24     6380

── Variable type: numeric ───────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate mean   sd   p0  p25  p50  p75 p100 hist
1 RATE                248         0.961 57.3 13.2 38.5 45.8 51.9 68.1 83.6 █▁▁▁▁
```

Figure 3: Summary after cleaning

So we see that in the summary, after cleaning the dataset, there are almost 248 missing values in it. We handle the missing values by applying linear interpolation using the code. And then, we plot the data to see what the data looks like.

```r
df$RATE <- na.approx(df$RATE, na.rm = FALSE)
ggplot(df, aes(x = DATE, y = RATE)) +
  geom_line() +
  labs(x = "Date",
       y = "Rate") +
  theme_minimal()
```

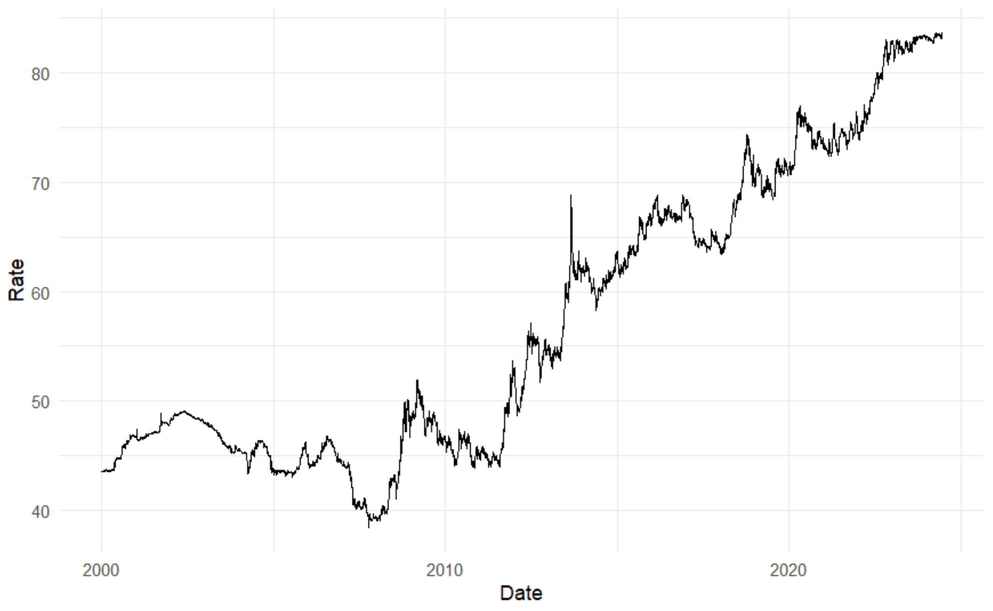


Figure 4: Plot of data

From the plot, we can see that there is an increasing trend in the graph, providing insight into the overall direction of the data. There are also considerable short-term fluctuations throughout the period, indicating volatility.

3.5 Data Analysis

3.5.1 ACF and PACF Plots

The Autocorrelation function[2] measures the linear relationship between lagged values of a time series. The equation is as follows:

$$r_k = \frac{\sum_{t=k+1}^{T}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2} = \frac{\text{Cov}(x_t, x_{t-k})}{\text{Var}(x_t)} \tag{1}$$

where:

- T is the length of the time series,
- $\bar{x}$ is the mean of the series,
- k is the lag.

The partial autocorrelation function (PACF) measures the correlation between time series observations separated by k time units, y_t and y_{t-k}, after removing the effects of all shorter lag correlations $y_{t-1}, y_{t-2}, \ldots, y_{t-(k-1)}$. The PACF plot is a graphical representation of the correlation of a time series with itself at different lags, after removing the effects of the previous lags.

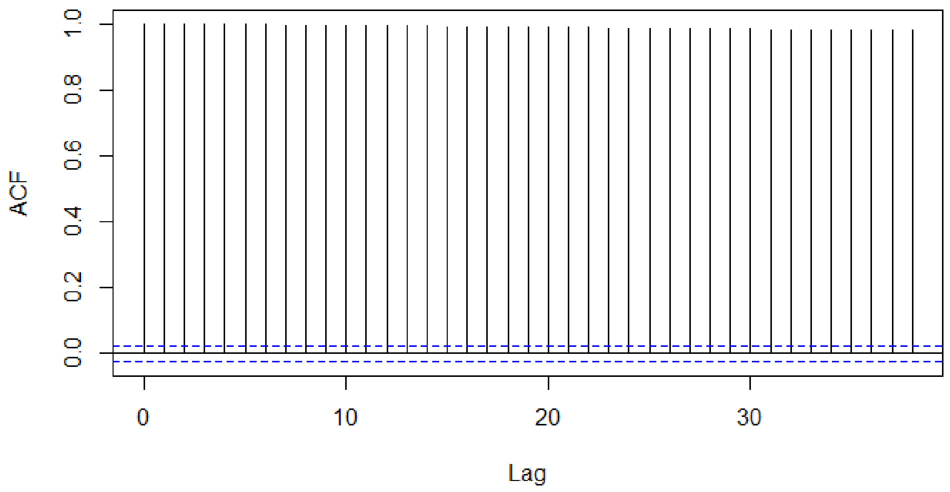


Figure 5: ACF plot

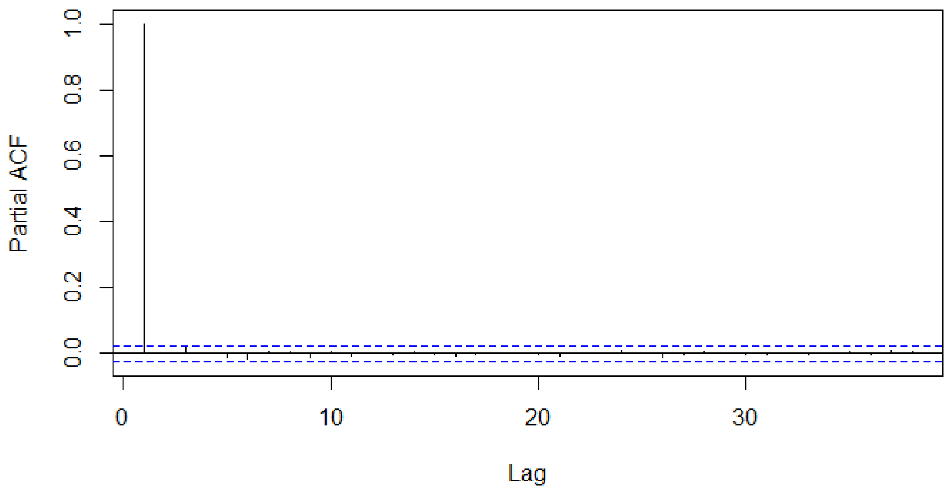


Figure 6: PACF plot

The ACF and PACF plots of the data are given above. As we can see, the ACF plot has significant auto-correlation at many lags that do not decay quickly, which are very high, that is, they are close to 1. There is also a slow decay in the ACF plot, which suggests that there is a trend component. The lack of cyclical pattern in ACF states that there is no seasonality in the data. The PACF plot shows a significant spike at lag 1 and then drops off dramatically. This is a characteristic of an AR(1) process.

3.5.2 Linear Model

Let's fit a linear model to the data. A linear model is used to describe the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the data.

```
lm_model <- lm(RATE ~ DATE, data = df)
df$fitted <- predict(lm_model)
ggplot(df, aes(x = DATE, y = RATE)) +
  geom_line() +
  geom_line(aes(y = fitted), color = "red") +
  labs(x = "Time", y = "RATE")
```

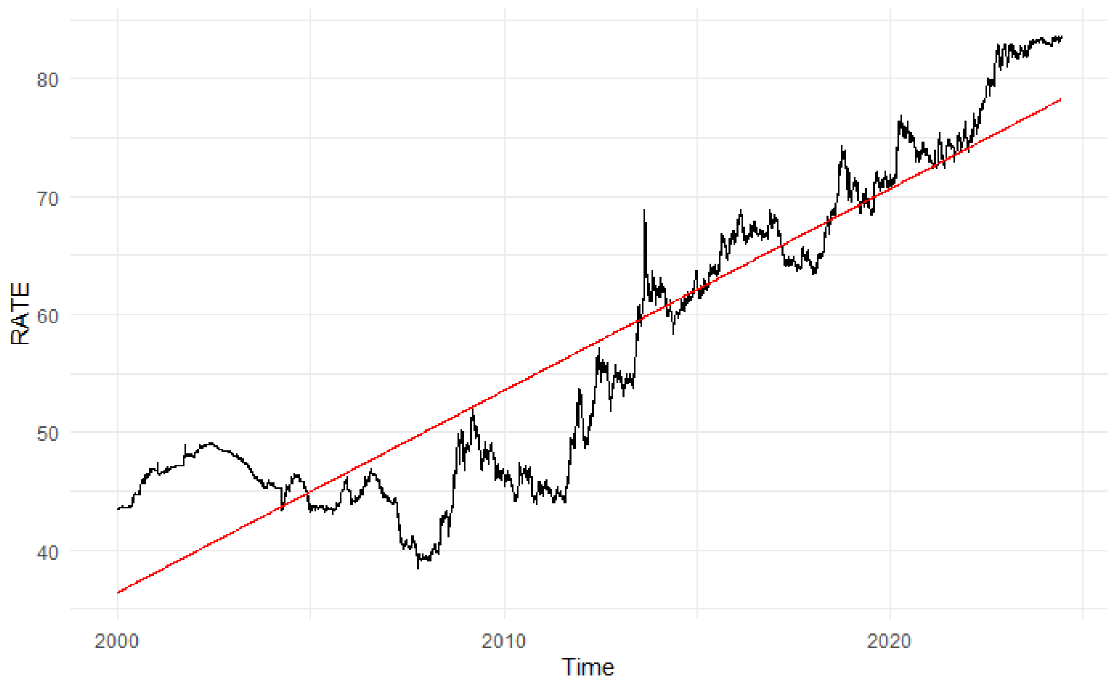


Figure 7: Linear model

In the figure the red line is the fitted values of the linear model, and the black one is the actual data. We can see from the figure itself, that the linear model is not fitting the data very well. Even though the linear model could capture the overall trend of the graph.

Now let us look at the residuals of the model [7]. We can check the residuals of the model by using the **residuals(model)** function in R. This function, when we pass the model as the parameter, will provide us the residuals or error of the model. This is calculated using the difference between the actual value and fitted values.

```
df$residuals <- residuals(lm_model)
ggplot(df, aes(x = DATE, y = residuals)) +
  geom_line() +
  labs(x = "Time", y = "Residuals") +
  theme_minimal()
```

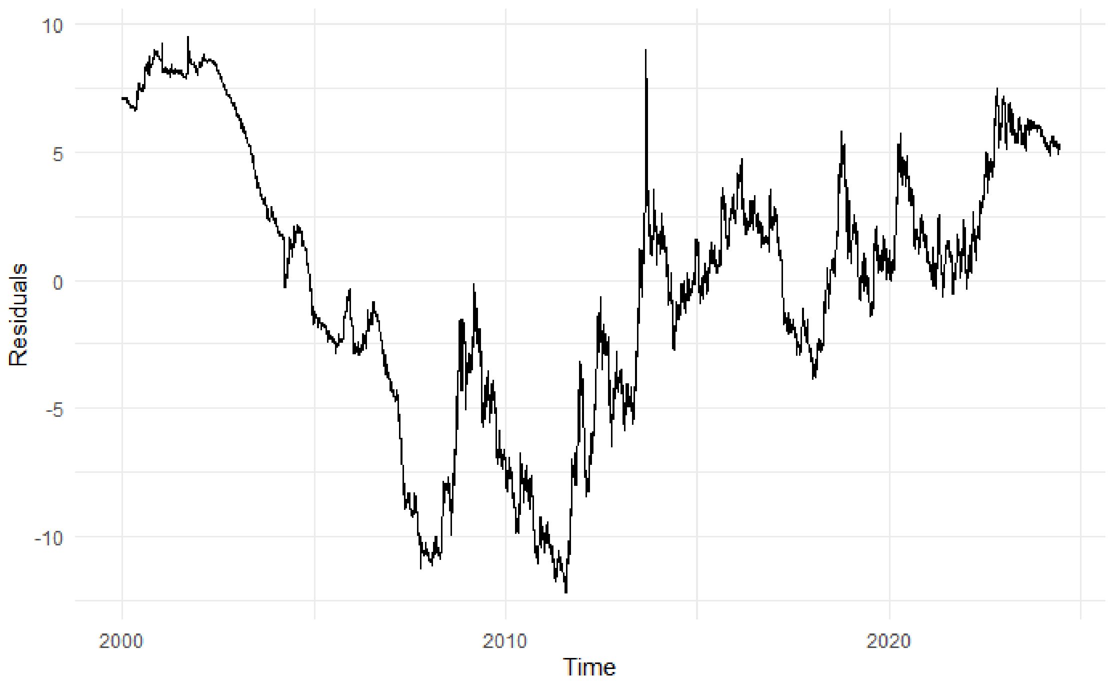


Figure 8: Residual plot of linear model

As we can see from the plot of the residuals [5], there is a significant difference between the fitted values and the original data. The residuals keep ranging from -10 to 10. The model specifically does not capture the early part of the data. This suggests that the linear model is not the best fit for the data.

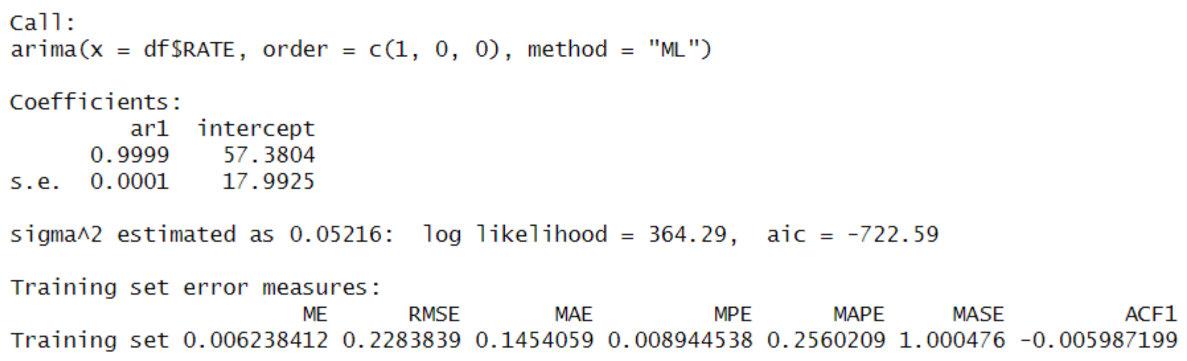
3.5.3 AR Model

Auto-Regressive models [2] are based on the idea that the current value of the series, y_t, can be explained as a function of p past values, $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ where p determines the number of steps into the past needed to forecast the current value. The general formula goes as follows:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{2}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ is independent identically distributed (iid) noise with mean 0 and variance σ^2, p represents the number of lag terms, and y_{t-i} is the i-th lagged value in the data.
Let us try fitting an AR(1) model on the data. For this, we will be using the **arima(x, order = c(p,0,0), method)** function, where x is the time series data, the p in the order parameter refers to the number of AR coefficients and for method parameter we can use the conditional sum of squares or maximum likelihood method. We here pass the data, specifying 1 AR coefficient which has to be found out by maximum likelihood method[3].

```
ar1 <- arima(df$RATE, order = c(1,0,0), method = "ML")
summary(ar1)
```



```
Call:
arima(x = df$RATE, order = c(1, 0, 0), method = "ML")

Coefficients:
         ar1  intercept
      0.9999    57.3804
s.e.  0.0001    17.9925

sigma^2 estimated as 0.05216:  log likelihood = 364.29,  aic = -722.59

Training set error measures:
                      ME      RMSE       MAE         MPE      MAPE     MASE         ACF1
Training set 0.006238412 0.2283839 0.1454059 0.008944538 0.2560209 1.000476 -0.005987199
```

Figure 9: AR(1) model

The coefficient of the AR(1) model is 0.9999, which is very near to one. This suggests that the series is following a unit root process that is non-stationary, and we know that it has a trend. So, we can say that the model does not fit the best for the data due to its near-unit root behavior.

3.5.4 ARIMA Model

An Auto-Regressive Integrated Moving Average (ARIMA) [2] model is specified by the following three parameters: (p, d, q).

Auto-Regressive (AR) part:

- The auto-regressive part involves regressing the variable on its own lagged (past) values.
- The parameter p is the number of lag observations included in the model (the number of terms in the auto-regressive part).

Integrated (I) part:

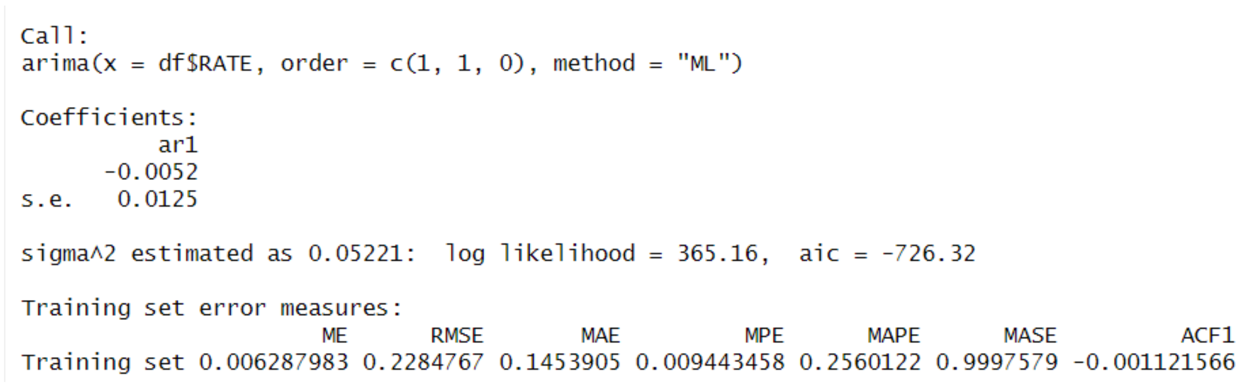
- The integrated part involves differencing the raw observations to make the time series stationary (i.e., having constant mean and variance over time).
- The parameter d is the number of times the differencing is applied to make the series stationary.

Moving Average (MA) part:

- The moving average part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past.
- The parameter q is the size of the moving average window (the number of terms in the moving average part).

Let us fit an ARIMA(1,1,0) model on our given data. We are differencing the data to remove the trend present in the data[3].

```
ar_d1 <- arima(df$RATE, order = c(1,1,0), method = "ML")
summary(ar_d1)
```

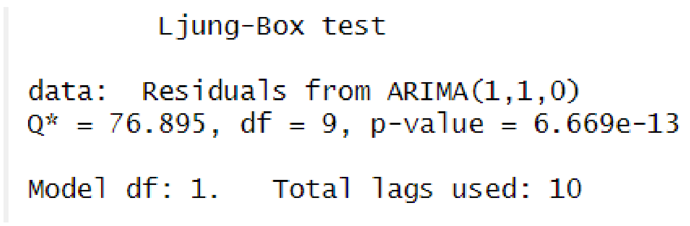


```
Call:
arima(x = df$RATE, order = c(1, 1, 0), method = "ML")

Coefficients:
          ar1
      -0.0052
s.e.   0.0125

sigma^2 estimated as 0.05221:  log likelihood = 365.16,  aic = -726.32

Training set error measures:
                      ME      RMSE       MAE         MPE      MAPE      MASE         ACF1
Training set 0.006287983 0.2284767 0.1453905 0.009443458 0.2560122 0.9997579 -0.001121566
```

Figure 10: ARIMA(1,1,0) model summary

As we can see, the model coefficient is -0.0052 value. This coefficient was found using the Maximum Likelihood Estimate method. Now let us check the residuals using **checkresiduals(model)** function in R, where we pass the model as a parameter.

```
checkresiduals(ar_d1)
```



```
	Ljung-Box test

data:  Residuals from ARIMA(1,1,0)
Q* = 76.895, df = 9, p-value = 6.669e-13

Model df: 1.   Total lags used: 10
```

Figure 11: Ljung Box Test of Residuals

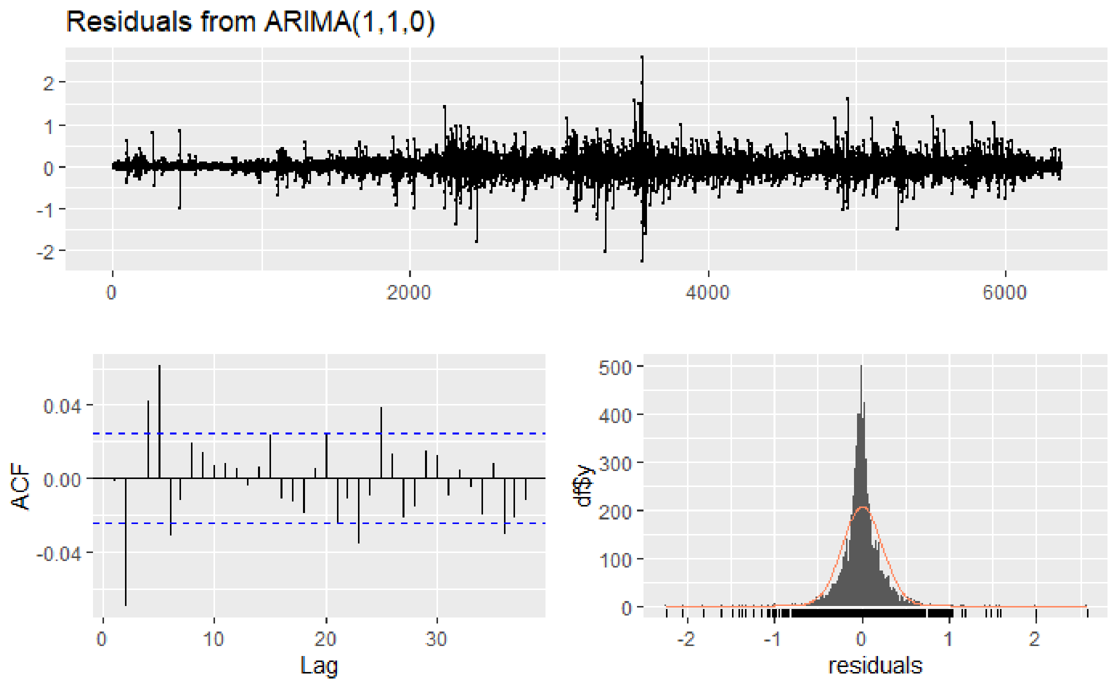


Figure 12: Different residual plots of ARIMA(1,1,0) model

The function gives the results as shown above. Let us go through it one by one. The top plot shows the plotting of the residuals. It provides us with an understanding of how well the model is fitted, as we see that most residuals are clustered around zero with some occasional spikes. The ACF plot of the residuals is provided in the bottom left of the figure. As we can see, most of the lags fall within the confidence interval, suggesting that there is little auto-correlation. The few values exceeding the confidence interval suggest that there is some internal structure that the model could not capture very well.

Ljung-Box [5] test is a test used in time series analysis to assess whether the lagged values of the residuals from a time series model exhibit significant auto-correlation up to a specified lag (null hypothesis). Here in the above result, we can assess the following:

- **Q***: It is the sum of squared auto-correlations of the residuals up to lag 10. This is the test static calculated for the Ljung-Box test.

- **df**: It represents the degrees of freedom, which is the difference in the number of lags used with a number of parameters. It determines the number of auto-correlations that are being tested.
- **p-value**: This is the probability associated with the test statistic. Here we have a very small p-value (e.g., 6.669×10^{-13}) suggests strong evidence to reject H_0, i.e. there is significant autocorrelation in the residuals.

As there is significant autocorrelation in the residuals, we can say that the model is not the best fit for the given time series data. So we should go for nonlinear models such as TAR, STAR, etc.

3.5.5 TAR Model

Threshold Autoregressive (TAR) model [2] is a type of nonlinear time series model that incorporates a strict threshold to capture regime changes in the data. In TAR models, the regime switches abruptly when the time series crosses a certain threshold.
The Threshold Autoregressive (TAR) model is defined as:

$$X_t = \begin{cases} \phi_{1,0} + \sum_{i=1}^{p_1} \phi_{1,i} X_{t-i} + \epsilon_{1,t} & \text{if } X_{t-d} \leq \gamma \\ \phi_{2,0} + \sum_{i=1}^{p_2} \phi_{2,i} X_{t-i} + \epsilon_{2,t} & \text{if } X_{t-d} > \gamma \end{cases} \tag{3}$$

where:

- X_t is the time series data.
- $\phi_{1,0}, \phi_{2,0}$ are the intercept terms for the two regimes.
- p_1, p_2 are the number of lagged observations included in the model for each regime.
- $\phi_{1,i}, \phi_{2,i}$ are the autoregressive coefficients for each regime.
- γ is the threshold value.
- $\epsilon_{1,t}, \epsilon_{2,t}$ are the white noise error terms for each regime.

We can fit a TAR model for the data in R using the `setar(x, m = 2, thDelay = 1, model = "TAR")` function from the `tsDyn` package in R, where:

- `x` is the time series data,
- `m` is the embedding dimension,
- `thDelay` is the time delay for the threshold variable,
- `model` is used to represent which model to use.

```r
tar_model <- setar(df$RATE, m = 2, thDelay = 1, model = "TAR")
summary(tar_model)
```

```
SETAR model ( 2 regimes)
Coefficients:
Low regime:
     const.L       phiL.1       phiL.2
-0.008769538  0.804014735  0.196357122

High regime:
     const.H       phiH.1       phiH.2
 0.000872988  1.010832919 -0.010743396

Threshold:
-Variable: Z(t) = + (0) X(t)+ (1)X(t-1)
-Value: 44.52
Proportion of points in low regime: 15.02%       High regime: 84.98%

Residuals:
       Min         1Q     Median         3Q        Max
-2.2845353 -0.0889303 -0.0052547  0.0879108  2.5319251

Fit:
residuals variance = 0.05194,  AIC = -18856, MAPE = 0.2562%

Coefficient(s):

           Estimate  Std. Error  t value  Pr(>|t|)
const.L -0.00876954  0.17692855  -0.0496    0.9605
phiL.1   0.80401473  0.04365901  18.4158 < 2.2e-16 ***
phiL.2   0.19635712  0.04378933   4.4841 7.449e-06 ***
const.H  0.00087299  0.01502571   0.0581    0.9537
phiH.1   1.01083292  0.01305200  77.4466 < 2.2e-16 ***
phiH.2  -0.01074340  0.01305536  -0.8229    0.4106
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Threshold
Variable: Z(t) = + (0) X(t) + (1) X(t-1)

Value: 44.52
```

Figure 13: TAR model summary

We used the **SETAR()** function to fit a TAR model[3], and we got the following model. The threshold value was found to be 44.52, and the estimates of each regime AR coefficients are given too. As we can see in the summary, only 15.02% is in the lower regime, and the rest, that is 84.98%, are in the higher regime. So, the residuals range from -2.28 to 2.53.

Now let us look into the fitting of the model. We shall be separating the fitted values based on the regime, and plotting the fitted values along the original data.

```
threshold <- 44.52

fitted_values <- fitted.values(tar_model)

df$Fitted_RATE <- c(rep(NA, length(df$RATE) - length(fitted_values)),
    fitted_values)

df$Regime <- ifelse(df$Fitted_RATE > threshold, 1, 2)

ggplot(df, aes(x = as.Date(DATE))) +
  geom_line(aes(y = RATE), color = "red") +
  geom_line(aes(y = Fitted_RATE, color = as.factor(Regime))) +
  scale_color_manual(values = c("1" = "blue", "2" = "green")) +
  geom_hline(yintercept = threshold, color = "black", size = 1) +
  labs(x = "Date",
       y = "RATE",
       color = "Regime") +
  theme_minimal() +
  theme(legend.position = "none")
```

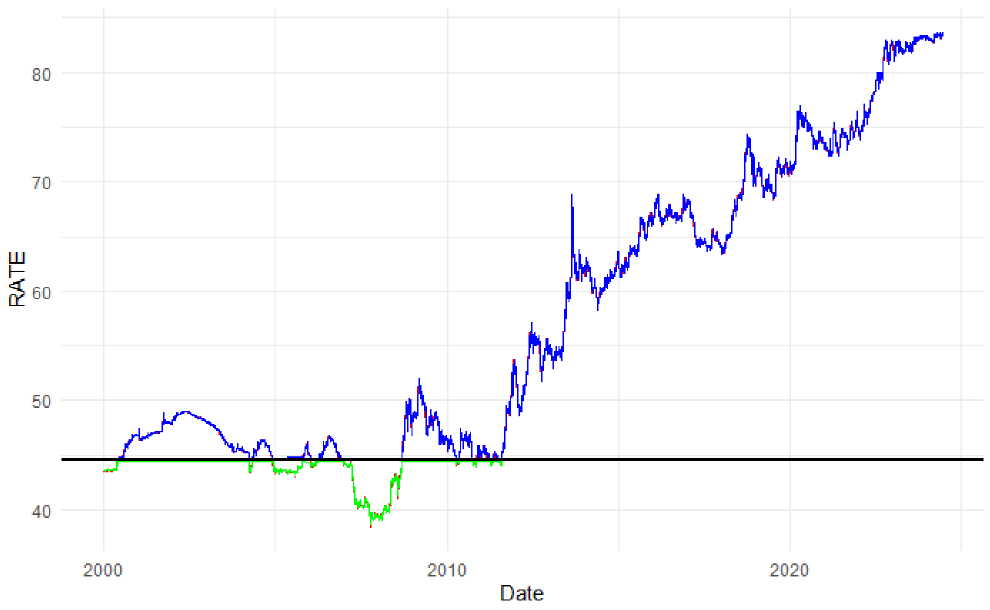


Figure 14: Fitting of TAR model

As we can see in the above graph, the fitted values almost precisely fit the original data. This proves that the given model is a good fit for the data. For further understanding, let us look into the ACF plot of residuals.

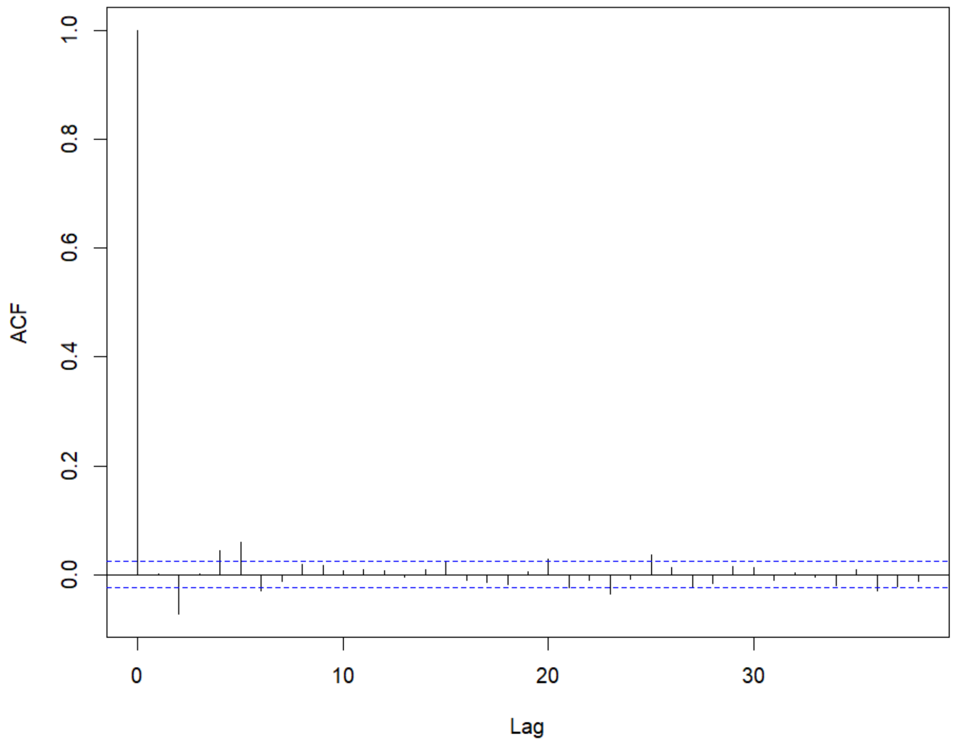


Figure 15: ACF of TAR model residuals

The residuals of the plot have a very low correlation, stating that the TAR model captured the complexities of the data well. There are only a few lags above the confidence interval. So, the TAR model is a good fit for the data. We shall be seeing further forecasting using the model in the results section.

3.5.6 STAR Model

Smooth Threshold Auto-Regressive (STAR) model [4] is a nonlinear time series model that incorporates smooth threshold functions to capture nonlinear relationships between the current observation and its lagged values. Unlike the hard threshold in Threshold Auto-Regressive (TAR) models, where the regime switches abruptly at a certain threshold, the smooth threshold function allows for gradual transitions between regimes. Common choices for S include logistic or exponential functions.
The Smooth Threshold Auto-Regressive (STAR) model is defined as:

$$X_t = \mu + \sum_{i=1}^{p} \phi_i \left(X_{t-i} - \gamma_i S(X_{t-i-1}) \right) + \epsilon_t \tag{4}$$

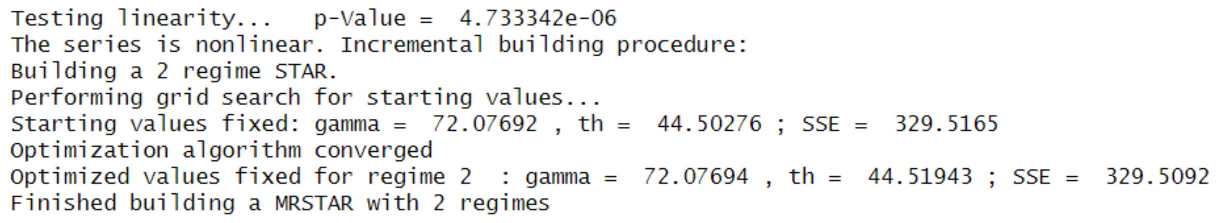
where:

- X_t is the time series at time t.
- μ is the intercept term.
- p is the number of lagged observations in the model.
- ϕ_i are the autoregressive coefficients.
- γ_i are the threshold parameters.
- $S(\cdot)$ is the smooth threshold function.
- ϵ_t is the white noise error term.

We can fit a STAR model for the data in R using the **star(x, m=2, noRegimes, d=1, thDelay = 1, sig = 0.05)** function from the tsDyn package in R, where:

- `x` is the time series data,
- `m` is the embedding dimension,
- `noRegimes` is the maximum number of regimes,
- `d` is the time delay,
- `thDelay` is the time delay for the threshold variable,
- `sig` is the significance level.

```
star_model <- star(df$RATE, m = 3, noRegimes = 2, d = 1, thDelay = 1,
   sig = 0.05)
```



```
Testing linearity...   p-Value =  4.733342e-06
The series is nonlinear. Incremental building procedure:
Building a 2 regime STAR.
Performing grid search for starting values...
Starting values fixed: gamma =  72.07692 , th =  44.50276 ; SSE =  329.5165
Optimization algorithm converged
Optimized values fixed for regime 2  : gamma =  72.07694 , th =  44.51943 ; SSE =  329.5092
Finished building a MRSTAR with 2 regimes
```

Figure 16: STAR model

As seen in the fitting of the STAR model, the starting value of gamma is 72.07692, which influences how sharply or gently the model's regime shifts occur. The higher the gamma value, the sharper the transition between regimes. The theta value is chosen as 44.50276, which determines the point around which the transition between regimes occurs.

We shall now check into the fitted values of the model. We can get the model's fitted values using the **fitted.values(model)** function in R by passing the model into it. After plotting the values alongside the original, we can see how well the model fits the data.

```r
fitted_values <- fitted.values(star_model)

combined_df <- data.frame(
  Date = df$DATE,
  Original_RATE = df$RATE,
  Fitted_RATE = c(rep(NA, length(df$RATE) - length(fitted_values)),
      fitted_values)
)

ggplot(combined_df, aes(x = Date)) +
  geom_line(aes(y = Original_RATE, color = "Original")) +
  geom_line(aes(y = Fitted_RATE, color = "Fitted")) +
  scale_color_manual(values = c("Original" = "blue", "Fitted" = "red"))
      +
  labs(x = "Date",
       y = "RATE",
       color = "Legend") +
  theme_minimal()
```

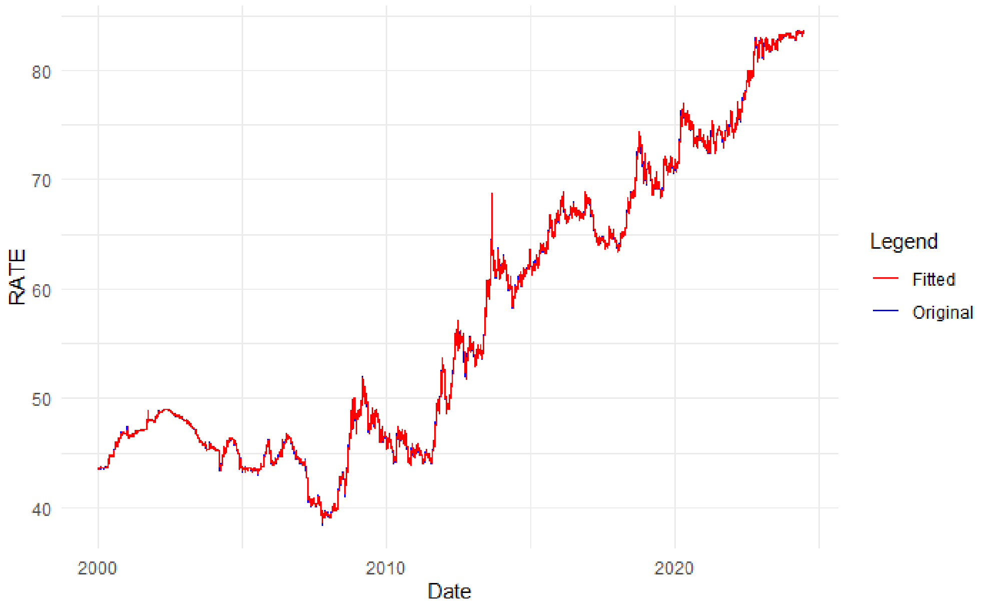


Figure 17: Fitted values of STAR model with original data

From the above graph, we can see that the fitted values of the STAR model fit the original data very well, as the original data values are overlapped very well. So we can infer that the model is a good fit for the data.

Let us look into the ACF plot of the model residuals to look into our inference further accurately. As said before we will be using the **ACF(data)** function to plot the ACF plot.

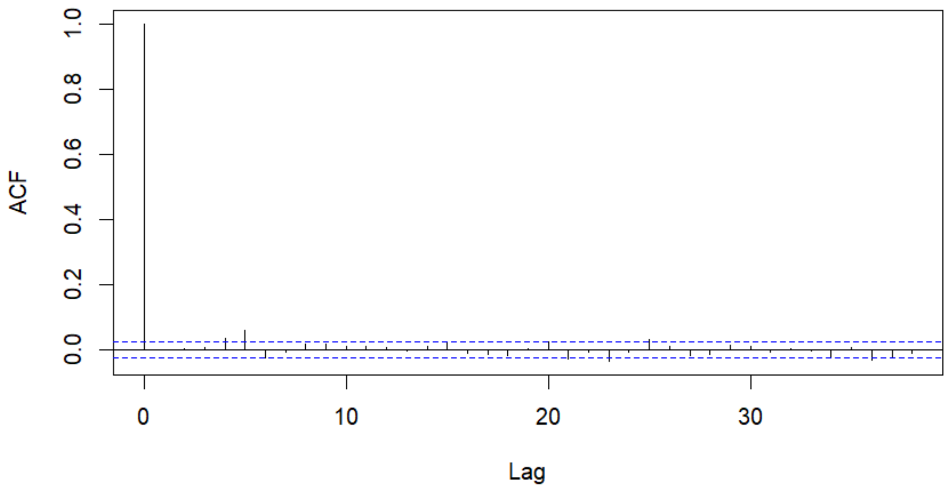


Figure 18: ACF of STAR model residuals

As we see from the plot above, there are fewer values that are crossing the confidence interval compared to the plot of the residuals of the TAR model. This shows that it has an even better understanding of the data than the TAR model. This model accurately fits the data. We shall further look into forecasting using the model in the results section.

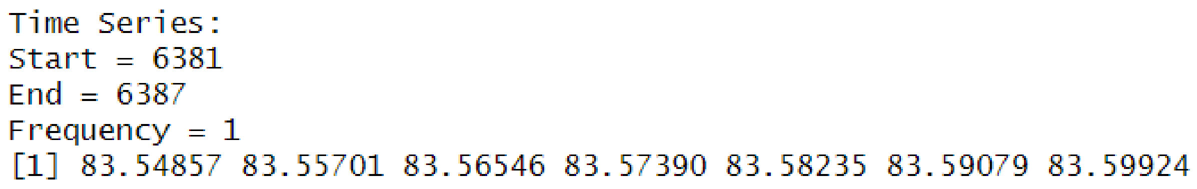
4 Results

We could see that both the TAR model and STAR model fitted the data the best compared to other linear models. Both the models fitted the data very well and the residuals of both models were very low. The residuals of the models had very little auto-correlation in different lags, too.

4.1 Forecast using TAR model

Let us use the **forecast(model, n.ahead=1)** function to predict future values. We pass the model and use the n.ahead parameter to specify how long the prediction should be. Then we shall be plotting the forecast along with the original data to visualize how the future prediction will be.

```
forecasts <- predict(tar_model, n.ahead = 7)
print(forecasts)
```



```
Time Series:
Start = 6381
End = 6387
Frequency = 1
[1] 83.54857 83.55701 83.56546 83.57390 83.58235 83.59079 83.59924
```

Figure 19: TAR Forecast Values

These are the predictions for the next 7 days using both the TAR and STAR models. We can see that both models provide almost similar results in prediction. Thus could state that both models fit the data almost similarly.

Now, let us look into the plotting of the predicted values by combining them with the original data.

```r
last_200_df <- tail(df, 200)

forecast_df <- data.frame(
  Date = seq(max(as.Date(df$DATE)) + 1, by = "day", length.out = 7),
  RATE = as.numeric(forecasts)
)

combined_df <- rbind(
  data.frame(Date = last_200_df$DATE, RATE = last_200_df$RATE, Type = "
     Original"),
  data.frame(Date = forecast_df$Date, RATE = forecast_df$RATE, Type = "
     Forecast")
)

ggplot(combined_df, aes(x = Date, y = RATE, color = Type)) +
  geom_line() +
  scale_color_manual(values = c("Original" = "blue", "Forecast" = "red"
     )) +
  labs(title= "STAR model forecast",
       x = "Date",
       y = "RATE",
       color = "Legend") +
  theme_minimal()
```

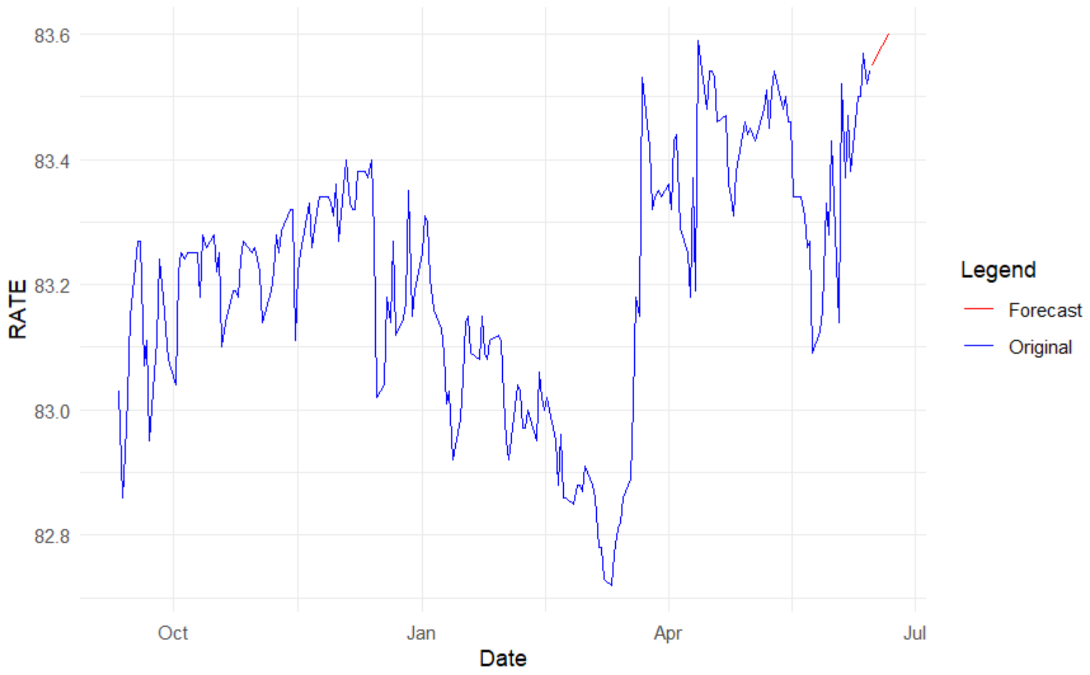


Figure 20: Forecasts of TAR model with original data

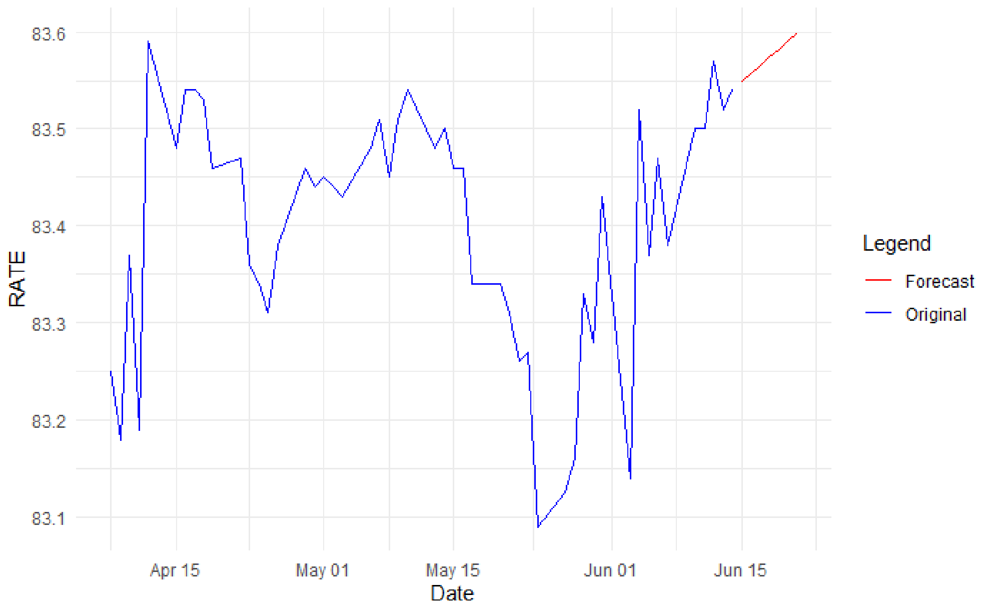
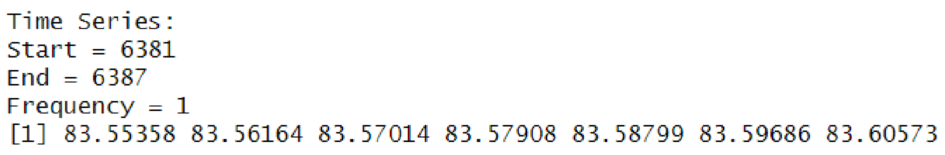


Figure 21: Zoomed-in TAR forecast

4.2 Forecast using STAR model

Similar to the above forecasting, we can do the same for the STAR model. We shall predict the next 7 forecasted values using the function and shall plot them.

```
forecasts <- predict(star_model, n.ahead = 7)
print(forecasts)
```



```
Time Series:
Start = 6381
End = 6387
Frequency = 1
[1] 83.55358 83.56164 83.57014 83.57908 83.58799 83.59686 83.60573
```

Figure 22: STAR model forecast values

For easy visualization, we opted only a few starting values of the original data.

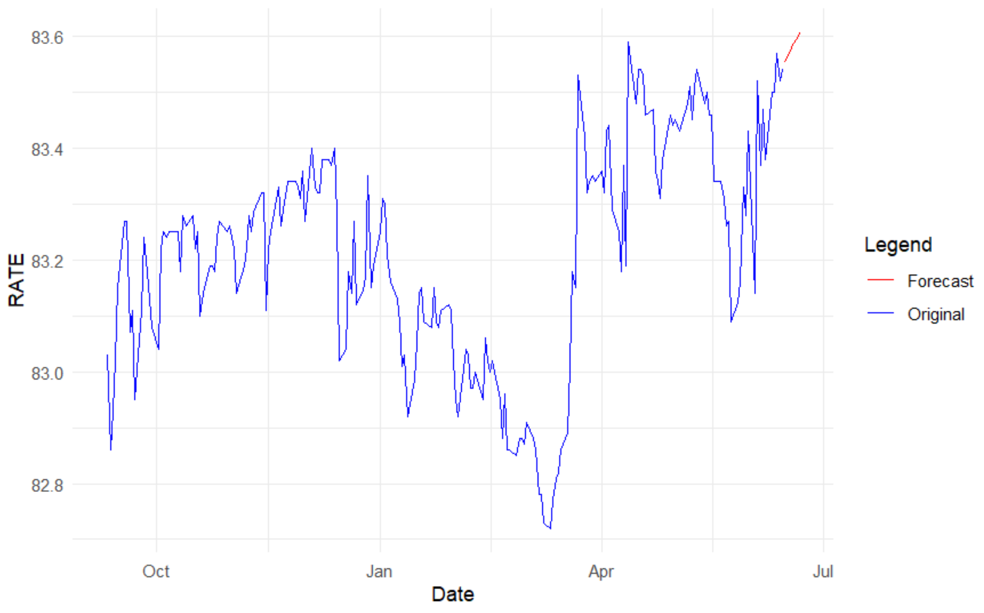


Figure 23: Forecasts of STAR model with original data

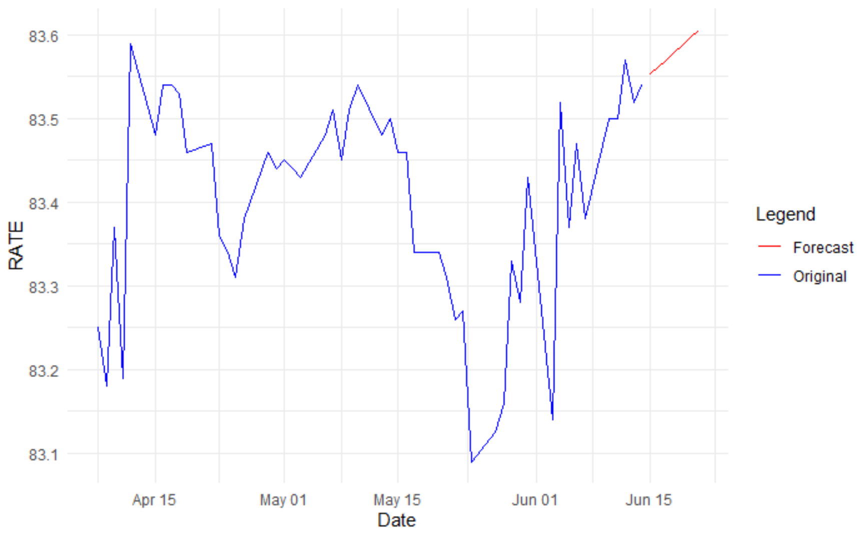


Figure 24: Zoomed-in STAR forecast

4.3 Model Evaluation

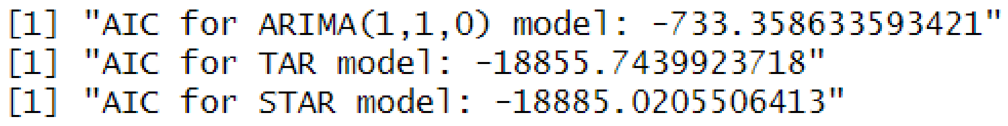
Here, we can see in the results that both of them give almost similar forecasts. This shows that both models capture the trend similarly. Thus we can say that both the models have very good fitting and forecasting of the data.

Now we can compare the models used using the Akaike Information Criterion (AIC) values of the models using the **AIC(model)** function in R. AIC is used for the evaluation of a model to know how well the model works. The lower the AIC value, the better the model is. The AIC is computed using the formula:

$$\text{AIC} = 2k - 2\ln(L) \tag{5}$$

where k is the number of parameters in the model, and $\ln(L)$ is the log-likelihood of the model.

```
aic_ar_d1_model <- AIC(ar_d1)
aic_tar_model <- AIC(tar_model)
aic_star_model <- AIC(star_model)

print(paste("AIC for ARIMA(1,1,0) model:", aic_ar_d1_model))
print(paste("AIC for TAR model:", aic_tar_model))
print(paste("AIC for STAR model:", aic_star_model))
```



```
[1] "AIC for ARIMA(1,1,0) model: -733.358633593421"
[1] "AIC for TAR model: -18855.7439923718"
[1] "AIC for STAR model: -18885.0205506413"
```

Figure 25: AIC values of the models

Now we can see that the STAR model has a lower AIC value compared to the TAR model, stating that the STAR model is the better model over the TAR model. For further inference, we can see the AIC value of the ARIMA(1,1,0) model also here. We can see how much better the AIC value of the non-linear models is compared to that of the ARIMA model.

5 Conclusion

This case study analyzed and forecasted daily prices of the INR-USD exchange rate from January 03, 2000, to June 16, 2024. Various time series models, including AR, ARIMA, and Threshold Autoregressive (TAR) models, were used and evaluated to capture the financial changes in the exchange rate. Linear models struggled to fit and capture the data relationship, whereas nonlinear models like TAR and STAR fit the data well. The STAR model provided the lowest AIC values, indicating it was superior to TAR for this dataset. The TAR model with two regimes closely matched actual data, while both models effectively predicted trends not captured by linear models. In conclusion, selecting an appropriate model is crucial for capturing data changes. Since most data is nonlinear, further research in nonlinear time series forecasting is needed. This study emphasizes the importance of nonlinear modeling for capturing complexity and enhancing prediction accuracy in financial forecasting.

6 References

[1] Federal Reserve Bank. (n.d.). Foreign Exchange Rates - H.10. Retrieved from `https://www.federalreserve.gov/releases/h10/current/default.htm`

[2] Tsay, R., & Chen, R. (Year). *Nonlinear Time Series Analysis*.

[3] Kratofil, M. A. (Year). *Introductory Time Series with R*.

[4] van Dijk, D., Teräsvirta, T., & Franses, P. H. (Year). Smooth Transition Autoregressive Models: A Survey of Recent Developments. Econometric Institute Research Report EI2000-23/A.

[5] Hyndman, R. J., & Athanasopoulos, G. (Year). *Forecasting: Principles and Practice* (2nd ed.)

[6] Holmes, E. E., Scheuerell, M. D., & Ward, E. J. (Year). *Applied Time Series Analysis for Fisheries and Environmental Sciences*

[7] Haben, S., Voss, M., & Holderbaum, W. (2023). Time Series Forecasting: Core Concepts and Definitions. In *Core Concepts and Methods in Load Forecasting*. Springer. `https://doi.org/10.1007/978-3-031-27852-5_5`

[8] Predicting Exchange Rate between US Dollar (USD) and Indian Rupee (INR): An Empirical Analysis using SARIMA Model(2024). *International Journal of Research in Finance and Management*, 7(1), 1–10. `https://doi.org/10.33545/26175754.2024.v7.i1a.283`