Time Series Final Project

Isabella Xue

2023-05-24

Rubrics

1. Problem statement
2. Assumptions/Hypotheses about data and/or modeling
3. Data properties (stationarity, correlations, data distribution) and Exploratory data analysis
4. Data processing (anomaly detection, cleansing and imputations) and transformations
5. Feature engineering
6. Proposed approaches (model) with justification and trade-offs, if any
7. Results (accuracy) and learnings from the methodology
8. Future work

1. Problem statement

Create a robust time series forecasting model to accurately predict the daily exchange rate between the Chinese Yuan (CNY) and the US Dollar (USD), based on historical data from January 4, 2010, to May 19, 2023.

Goal: This model should be able to identify patterns and trends in the data in order to generate reliable predictions for future exchange rates and adjust predictions in the event of unexpected fluctuations and unforeseen circumstances.

Significance: Forecasting the CNY exchange rate provides valuable information to businesses/individuals engaged in import/export activities, helping them manage currency risks and make informed investment decisions.

2. Assumptions/Hypotheses

Assumptions:

- Independence: The observations are independent of each other, which means that the value of the exchange rate on a given day is not influenced by the values on preceding or subsequent days.
- Model Stability: Forecasting models assume that the relationships and patterns observed in the historical data will remain stable and hold true in the future. This assumption implies that the factors influencing the target variable will not undergo significant changes or structural shifts.

Hypotheses:

- Efficient Market Hypothesis: The exchange rate reflects all available information and adjusts instantaneously to new information.
- Autocorrelation: The exchange rate exhibits autocorrelation, meaning that the current value of the exchange rate is related to its previous values.

3. Data processing (anomaly detection, cleansing and imputations) and transformations

```r
#df <- read.csv('FRB_H10.csv', skip=1) %>% rename(Date = Time.Period, CNY = RXI_N.B.CH)
df <- read.csv('FRB_H10.csv', skip=1) %>% rename(Date = Time.Period, CNY = RXI_N.B.CH) %>% dplyr::filte

#df <- read.csv('euro-daily-hist_1999_2022.csv') %>% select(Period.Unit., X.US.dollar..) %>% rename(Dat

head(df)
```

```
##         Date    CNY
## 1 2010-01-01     ND
## 2 2010-01-04 6.8273
## 3 2010-01-05 6.8258
## 4 2010-01-06 6.8272
## 5 2010-01-07 6.8280
## 6 2010-01-08 6.8274
```

Anomaly detection, cleansing:

```r
# Check for duplicated rows based on the "Date" column
duplicated_rows <- anyDuplicated(df$Date)
cat("There is", duplicated_rows, "duplicated rows in Date column \n")
```

```
## There is 0 duplicated rows in Date column
```

```r
# Convert the "Date" column to Date format
df$Date <- as.Date(df$Date)

# Check for missing values in the "CNY" column
#CNY = ND is missing value
missing_values <- sum(is.na(df$CNY))
cat("There is", missing_values, "values in CNY column that is null \n")
```

```
## There is 0 values in CNY column that is null
```

```r
table(df$CNY == "ND")
```

```
##
## FALSE  TRUE
##  3348   143
```

```r
#143 missing values in CNY

#Remove the rows where CNY equals to ND
df <- df[df$CNY != 'ND', ]

# Check for consecutive dates and print missing dates
all_dates <- seq(min(df$Date), max(df$Date), by = "day")
missing_dates <- setdiff(all_dates, df$Date)
cat("There are", length(missing_dates), "days missing from the data")
```

```
## There are 1536 days missing from the data
```

```r
# Create a new data frame with the complete sequence of dates
complete_df <- data.frame(Date = all_dates)

# Merge the complete data frame with the original data frame, keeping only the missing dates
```

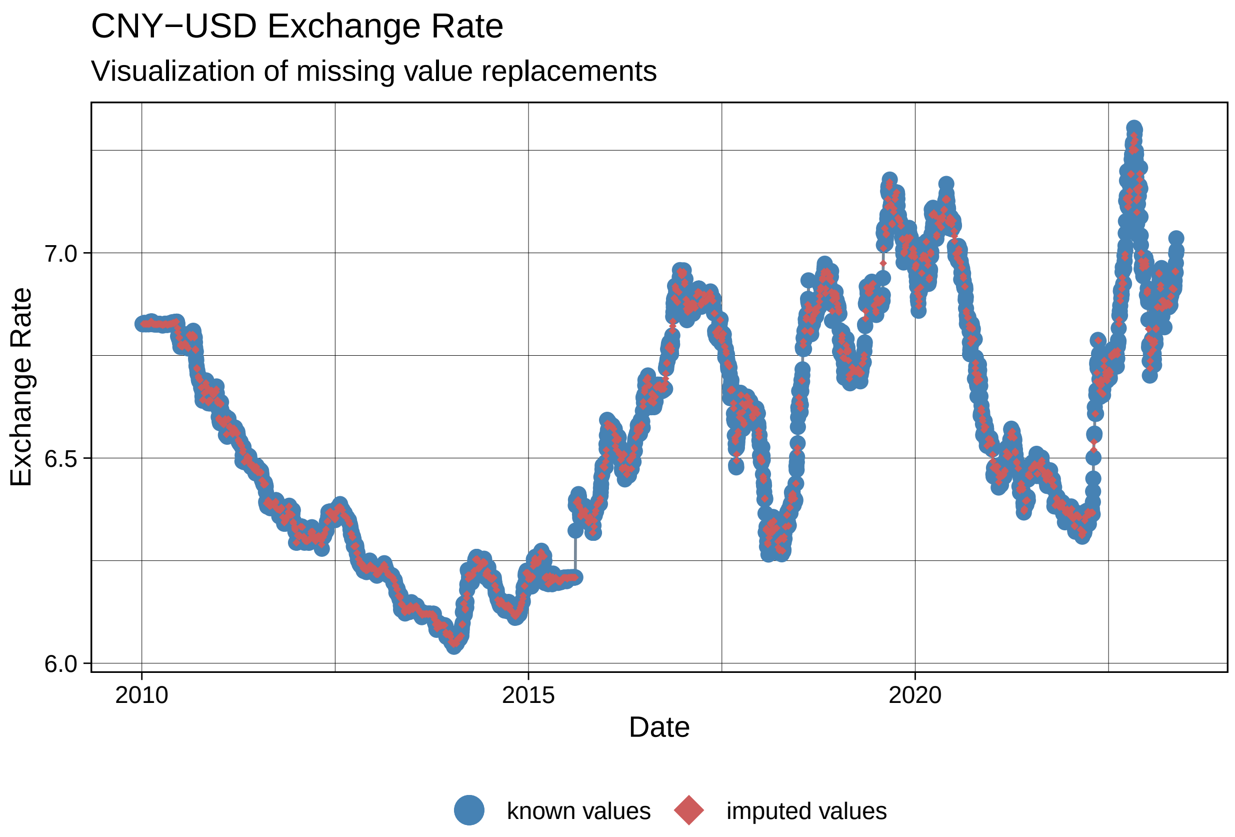
```r
missing_dates_df <- merge(complete_df, df, by = "Date", all.x = TRUE)

# Set CNY values to null for missing dates
missing_dates_df$CNY[is.na(missing_dates_df$CNY)] <- NA
```

Imputations and Data Transformation:

```r
ts_data <- ts(as.double(missing_dates_df$CNY))
ts_data_interp <- na.interp(ts_data, lambda = "auto")

ggplot_na_imputations(x_with_na = ts(as.double(missing_dates_df$CNY)), x_with_imputations = ts_data_int
                      title = "CNY-USD Exchange Rate",
                      x_axis_labels = all_dates,
                      xlab = "Date", ylab = "Exchange Rate",
                      size_points = 2.5,
                      size_imputations = 1)
```



```r
#Final dataframe
df_interp <- missing_dates_df %>% mutate(CNY = ts_data_interp)
```

4. Data properties (stationarity, correlations, data distribution) and Exploratory data analysis

```r
#stationarity
ts_obj <- ts(df_interp$CNY, start = c(2010, as.numeric(format(all_dates[1], "%j"))), frequency = 365)
adf_test <- adf.test(ts_obj)
kpss_test <- kpss.test(ts_obj)
```

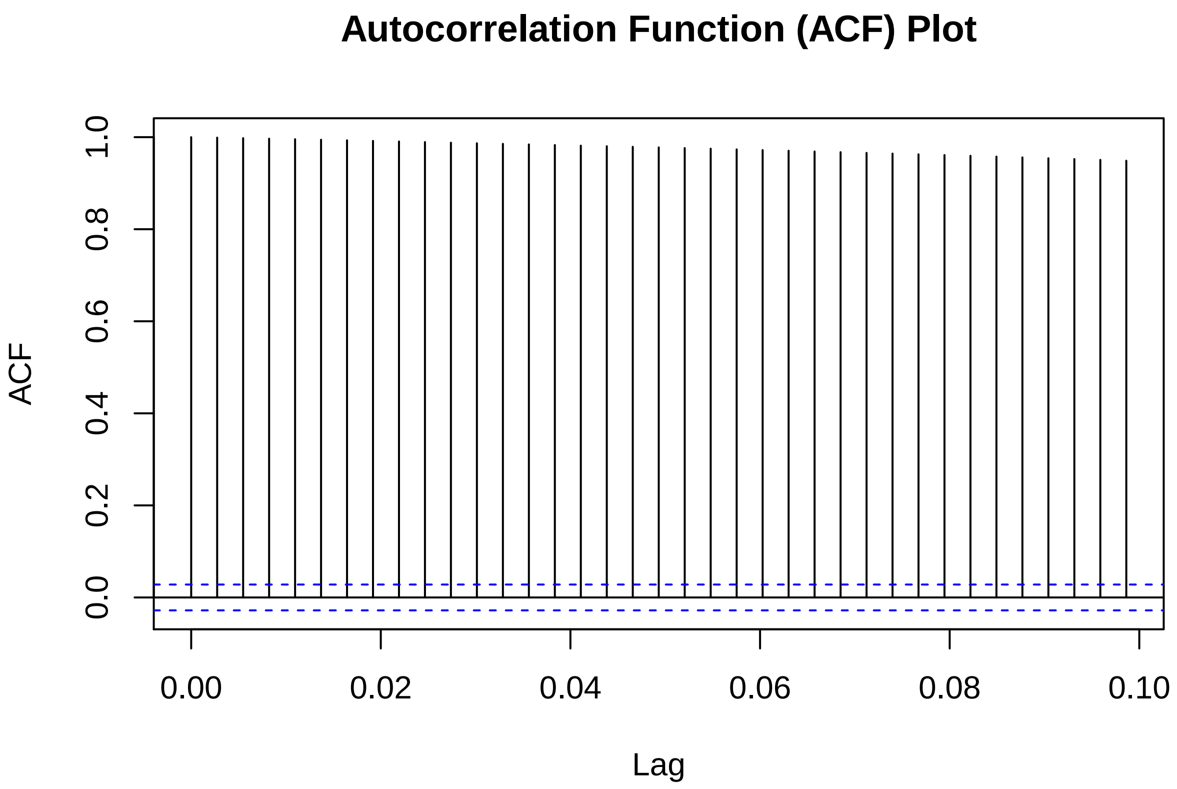
```r
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_obj
## Dickey-Fuller = -2.1604, Lag order = 16, p-value = 0.5104
## alternative hypothesis: stationary
```

```r
print(kpss_test)
```

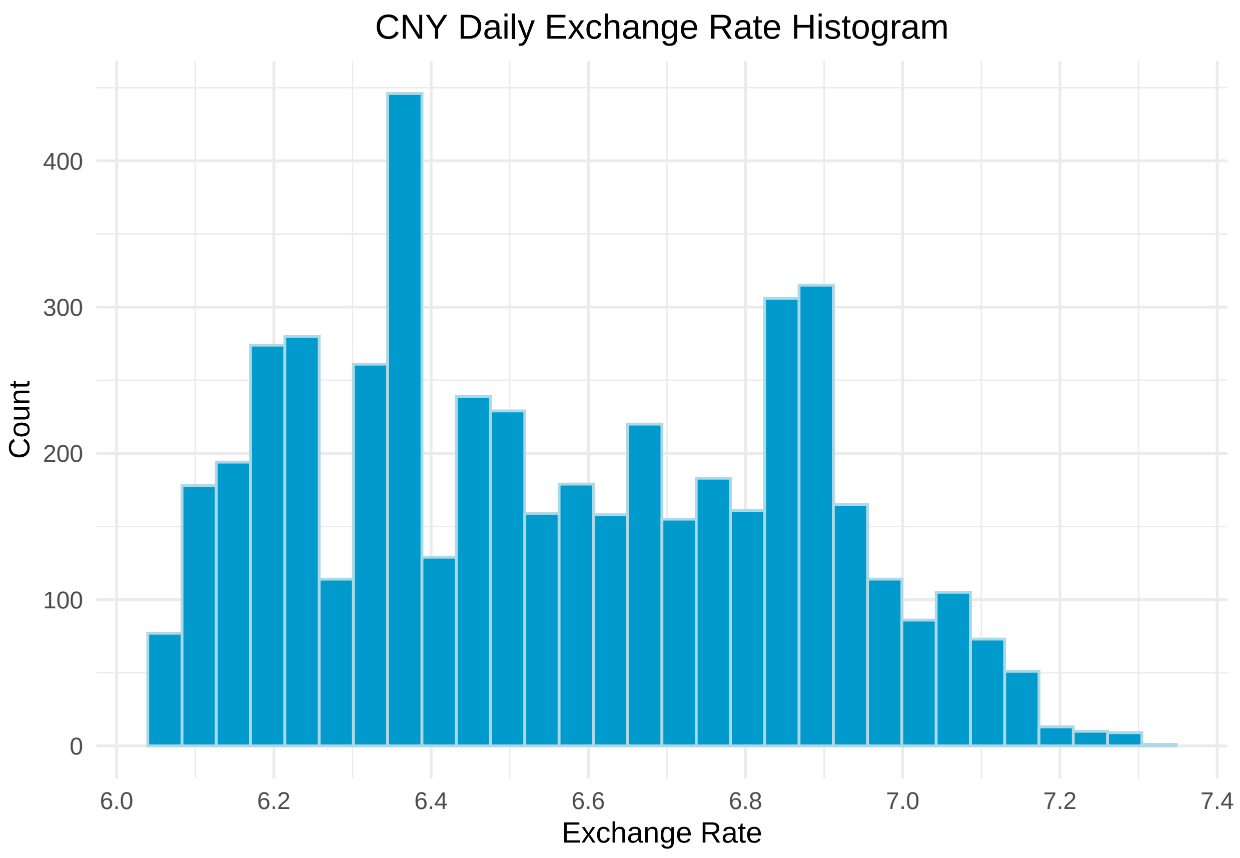
```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_obj
## KPSS Level = 13.219, Truncation lag parameter = 10, p-value = 0.01
```

```r
#autocorrelations
acf(ts_obj, main = "Autocorrelation Function (ACF) Plot")
```

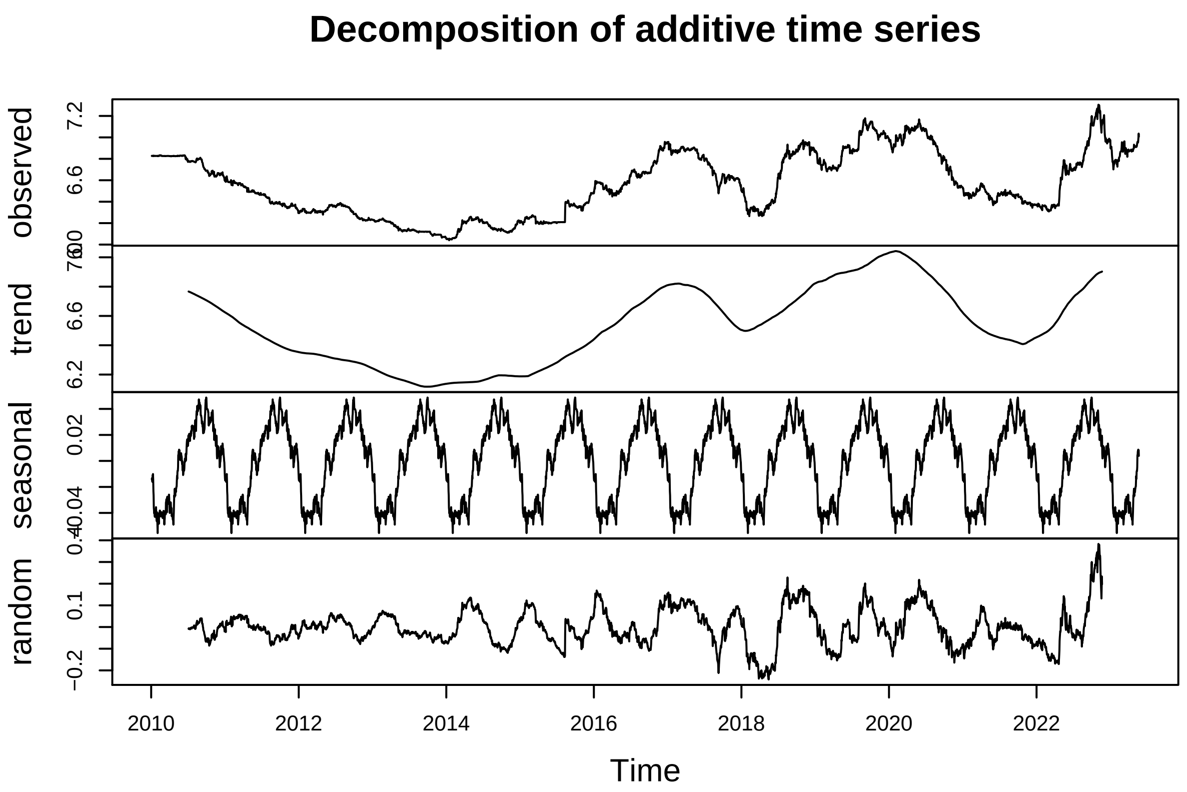


```r
#Distribution
ggplot(df_interp, aes(x=CNY)) +
  geom_histogram(fill="deepskyblue3", color="lightblue") +
  labs(title="CNY Daily Exchange Rate Histogram",x="Exchange Rate", y = "Count") +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks=scales::pretty_breaks(10))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#trend, seasonality
decomposition <- decompose(ts_obj)
plot(decomposition)
```



I observe:

- ADF and KPSS shows that CNY exchange rate time series is non-stationary.
- Acf plot indicates high autocorrelation.
- Distribution does not show specific pattern.
- Trend and high seasonality.

5. Feature Engineering

```
# Create a time series object for training and testing
train_ts <- head(ts_obj, -50)

test_ts <- tail(ts_obj, 50)
```

6. Proposed Approaches

Holt-Winters

```
hw_model <- HoltWinters(train_ts)
summary(hw_model)
```

```
##              Length Class  Mode
## fitted       17876  mts    numeric
## x             4834  ts     numeric
## alpha            1  -none- numeric
## beta             1  -none- numeric
## gamma            1  -none- numeric
## coefficients   367  -none- numeric
## seasonal          1   -none- character
## SSE               1   -none- numeric
## call              2   -none- call
```

```
hw_forecast <- forecast(hw_model, h = length(test_ts))
hw_mse <- mean((hw_forecast$mean - test_ts)^2)
hw_rmse <- sqrt(mean((c(hw_forecast$mean) - c(test_ts))^2))
hw_mape <- mean(abs((c(hw_forecast$mean) - c(test_ts)) / test_ts)) * 100

print(paste("Holt-Winters forecast MSE:", hw_mse))
```

```
## [1] "Holt-Winters forecast MSE: 0.00506994669479688"
```

```
print(paste("Holt-Winters forecast RMSE:", hw_rmse))
```

```
## [1] "Holt-Winters forecast RMSE: 0.07120355816107"
```

```
print(paste("Holt-Winters forecast MAPE:", hw_mape, "%"))
```

```
## [1] "Holt-Winters forecast MAPE: 0.782650189366689 %"
```

SARIMA:

```
# Fit the SARIMA model to the differenced series
sarima_model <- auto.arima(train_ts)
summary(sarima_model)
```

```
## Series: train_ts
## ARIMA(3,1,1)
##
## Coefficients:
##          ar1      ar2      ar3      ma1
##       1.0096  -0.0003  -0.0336  -0.9578
## s.e.  0.0208   0.0205   0.0149   0.0149
##
## sigma^2 = 0.000116:  log likelihood = 15042.8
## AIC=-30075.61   AICc=-30075.6   BIC=-30043.19
##
## Training set error measures:
##                        ME       RMSE         MAE         MPE       MAPE
## Training set 5.783217e-06 0.01076423 0.006100379 5.16046e-05 0.09171712
##                    MASE         ACF1
## Training set 0.02255017 5.093097e-05
```

```
# Generate forecasts using the SARIMA model
sarima_forecast <- forecast(sarima_model, h = length(test_ts))

# Evaluate the model's performance
sarima_mse <- mean((sarima_forecast$mean - test_ts)^2)
sarima_rmse <- sqrt(mean((c(sarima_forecast$mean)-c(test_ts))^2))
sarima_mape <- mean(abs((c(sarima_forecast$mean)-c(test_ts)) / test_ts)) * 100

# Print the evaluation metrics
print(paste("SARIMA forecast MSE:", sarima_mse))
```

```
## [1] "SARIMA forecast MSE: 0.00328187987784043"
```

```
print(paste("SARIMA forecast RMSE:", sarima_rmse))
```

```
## [1] "SARIMA forecast RMSE: 0.0572876939476572"
```

```
print(paste("SARIMA forecast MAPE:", sarima_mape, "%"))
```

```
## [1] "SARIMA forecast MAPE: 0.599304251487537 %"
```

ARFIMA

```
arfima_model <- forecast::arfima(train_ts)
summary(arfima_model)
```

```
##
## Call:
##   forecast::arfima(y = train_ts)
##
## Coefficients:
##          Estimate Std. Error z value Pr(>|z|)
## d       0.4998600  0.0005366  931.45   <2e-16 ***
## ma.ma1 -0.7658450  0.0086317  -88.72   <2e-16 ***
## ma.ma2 -0.6985252  0.0093270  -74.89   <2e-16 ***
## ma.ma3 -0.5712030  0.0084679  -67.45   <2e-16 ***
## ma.ma4 -0.4097131  0.0086033  -47.62   <2e-16 ***
## ma.ma5 -0.2314758  0.0083384  -27.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## sigma[eps] = 0.01255552
## [d.tol = 0.0001221, M = 100, h = 0.0001507]
## Log likelihood: 1.43e+04 ==> AIC = -28582.69 [7 deg.freedom]
```

```
arfima_forecast <- forecast::forecast(arfima_model, h = length(test_ts))
arfima_mse <- mean((arfima_forecast$mean - test_ts)^2)
arfima_rmse <- sqrt(mean((c(arfima_forecast$mean) - c(test_ts))^2))
arfima_mape <- mean(abs((c(arfima_forecast$mean) - c(test_ts)) / test_ts)) * 100

print(paste("ARFIMA forecast MSE:", sarima_mse))
```

```
## [1] "ARFIMA forecast MSE: 0.00328187987784043"
```

```
print(paste("ARFIMA forecast RMSE:", arfima_rmse))
```

```
## [1] "ARFIMA forecast RMSE: 0.0912136775054407"
```

```
print(paste("ARFIMA forecast MAPE:", arfima_mape, "%"))
```

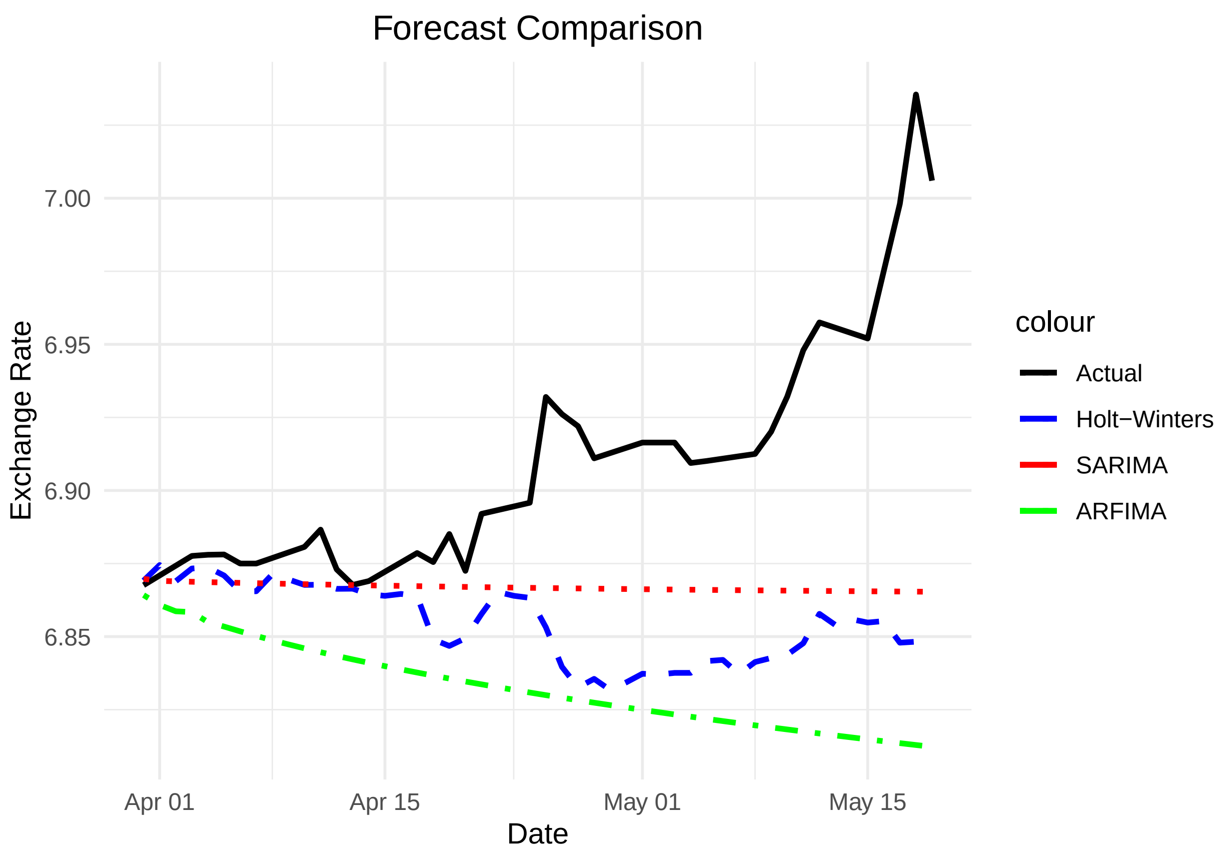
```
## [1] "ARFIMA forecast MAPE: 1.09209836546765 %"
```

7. Results and learnings from the methodology

Visualizing Actual vs. Forecast:

```
results_df <- data.frame(Date = tail(df_interp$Date, 50), Actual = as.vector(test_ts),
                 HoltWinters = as.vector(hw_forecast$mean),
                 SARIMA = as.vector(sarima_forecast$mean),
                 ARFIMA = as.vector(arfima_forecast$mean))
```

```r
# Create the plot
ggplot(data = results_df, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual"), size = 1) +
  geom_line(aes(y = HoltWinters, color = "Holt-Winters"), size = 1, linetype = "dashed") +
  geom_line(aes(y = SARIMA, color = "SARIMA"), size = 1, linetype = "dotted") +
  geom_line(aes(y = ARFIMA, color = "ARFIMA"), size = 1, linetype = "dotdash") +
  labs(x = "Date", y = "Exchange Rate", title = "Forecast Comparison") +
  scale_color_manual(values = c("Actual" = "black", "Holt-Winters" = "blue",
                                "SARIMA" = "red", "ARFIMA" = "green")) +
  theme_minimal() + theme(plot.title = element_text(hjust = 0.5))
```



8. Future Work

Limitations:

- Interpolation estimates missing values based on nearby observations, but it can't capture long-term trends and patterns in the data if the missing values are spread out over time. This can result in a loss of info and can affect accuracy of long-term forecasts.

Future works:

- Incorporating relevant external factors, such as economic indicators, interest rates, or geopolitical events, into the forecasting models
- Exploring more advanced time series models beyond SARIMA, ARFIMA, and Holt-Winters
- Extending the analysis to real-time forecasting, where new data is continuously incorporated into the models