**Project Title:**

**<AUTHOR NAME>**

# Abstract

The NIFTY dataset, which offers useful data for doing financial analysis, stock price forecasts, and operational efficiency evaluation, is the main subject of this study. To evaluate the dataset and produce future stock market value projections is the key goal.

Stock price forecasts based on past data are made using time series techniques like ARIMA and regression models.. The main outcomes include coming up with a model to predict future volume prices. The results indicate that the NIFTY dataset might be a useful tool for understanding market movements and making educated investing decisions.

# Introduction

The top 50 businesses listed on the National Stock Exchange make up the NIFTY, an index of the Indian stock market. It gives information on general market trends and acts as a benchmark for the Indian equity market. Companies from a range of industries are included in the index, which is frequently rebalanced. In order to gauge market sentiment, monitor sector performance, and make wise investment decisions, traders and investors closely monitor the NIFTY.

The NIFTY dataset was my first choice since it offers useful information for undertaking a range of analyses, such as financial analysis, stock price forecasts, and operational efficiency evaluation. My study' main goal is to examine this dataset and produce predictions for future stock market values.

I can do financial analysis using the NIFTY dataset to assess the company's financial performance over time. In order to spot trends and patterns, this research may entail looking at important financial data including sales, profits, expenses, and other pertinent metrics.

## Preprocessing

To archive my goal, I started the analysis by importing the required libraries into my R environment. I further, imported my dataset into my environment and converted the dataset into an R data object. The dataset had irregular intervals, therefore I converted the dataset into a monthly-based dataset for my time series analysis and later into a time series object.

The method of transforming the dataset into a monthly-based dataset produced a dataset that is not stationary. Using Dickey-Fuller Test, I confirmed that the p-value is greater than 0.05 which suggested that the time series object is non-stationary.

## Transformation

To stabilize the variance and hence make the time series object stationary, I had to implement several steps. These includes

1. Stabilizing the variance by using log
2. Fitting linear regression model
3. Applying Box-Cox transformation

I went further and performed decomposition in order to remove linear trends on the vwap pricing of the dataset. This finally gave me the results that I wanted. I performed another ADF test to confirm if my p-value has decreased to any value less than 0.05.

The ADF (Augmented Dickey-Fuller) test revealed that the time series item is stationary after performing the procedures stated above. The ADF test, which generated a p-value of 0.01 for this result, provided additional support.

Finally, I plotted the ACF and PCF in order to find out the dependency structure. These plots helped me to realize that ARIMA (1, 1, 1) would fit to provide accurate ADANIPORTS vwap.

## Conclusion

The results of this study demonstrate the importance of time series analysis as an effective tool for forecasting stock fluctuations and assisting with well-informed investment decisions. By using time series, one can get important insights into the patterns and trends present in stock data.

# Analysis

## Library and Data Preparation

I loaded below libraries in order to help in the analysis of the dataset



## Data Preprocessing

This involved reading the dataset from file and cleaning the datasets in order to remove unwanted rows and columns. I also tried to convert target columns into appropriate data types as show below.

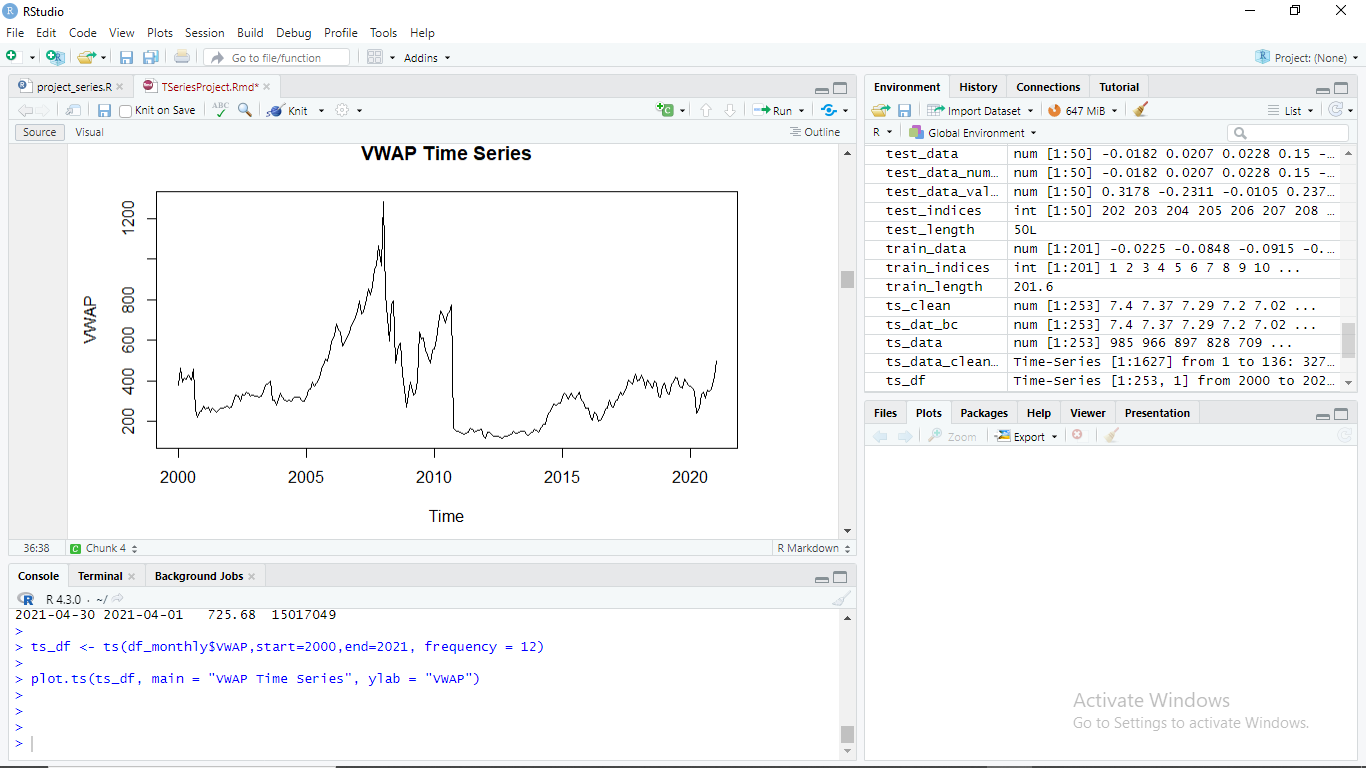


I went further to convert the data frame into a time series object. Since I needed a monthly-based data frame, I used the xts library to transform the dataset into a monthly based dataset.



## Plotting





From the above plot, the variance keeps on changing. This suggests that we may have strong seasonality. The plot does not show any increasing or decreasing trend.

Using Volume Prices



This plot using volume prices shows an increasing trend. Though from 2000, the trend does not seem to increase, there seem to be an increasing trend from 2008 to 2021. I will use the volume for further analysis.

Diagnosis

It is clear from the plot above that our time series item has non-stationarity. I utilized the Dickey-Fuller test, a commonly used statistical technique to detect whether there is a unit root in a time series item, to support this observation.



Our time series object's non-stationarity is further confirmed by the test findings mentioned above. The fact that the p-value (0.99) is higher than the usual significance level of 0.05 serves as a clue. As a result, we can say that the series displays non-stationarity.

## Transformation

The steps below were followed to make the time series object stationary

Step 1. Model Fitting

Step 2. Stabilizing variance using logs

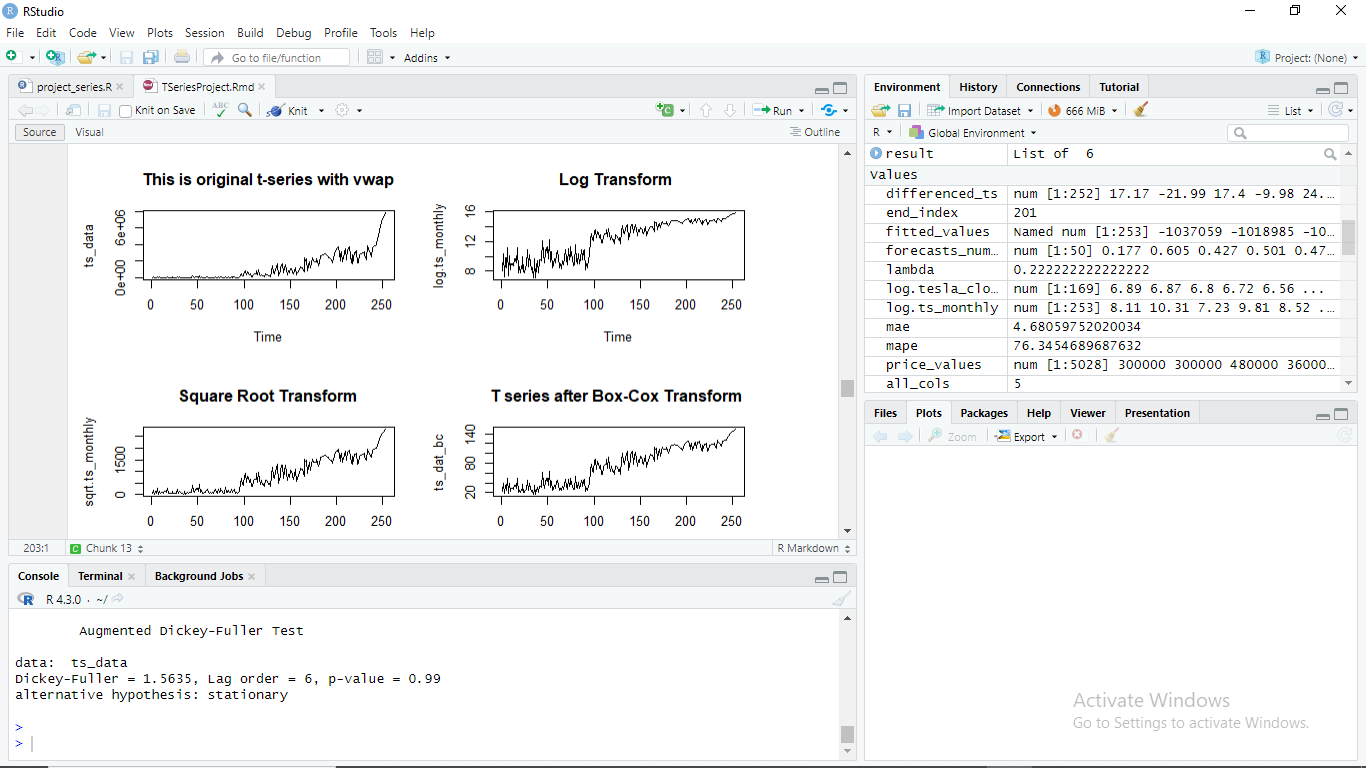
Step 3. Applying Box-Cox transformation

Step 4.Performing power transformation

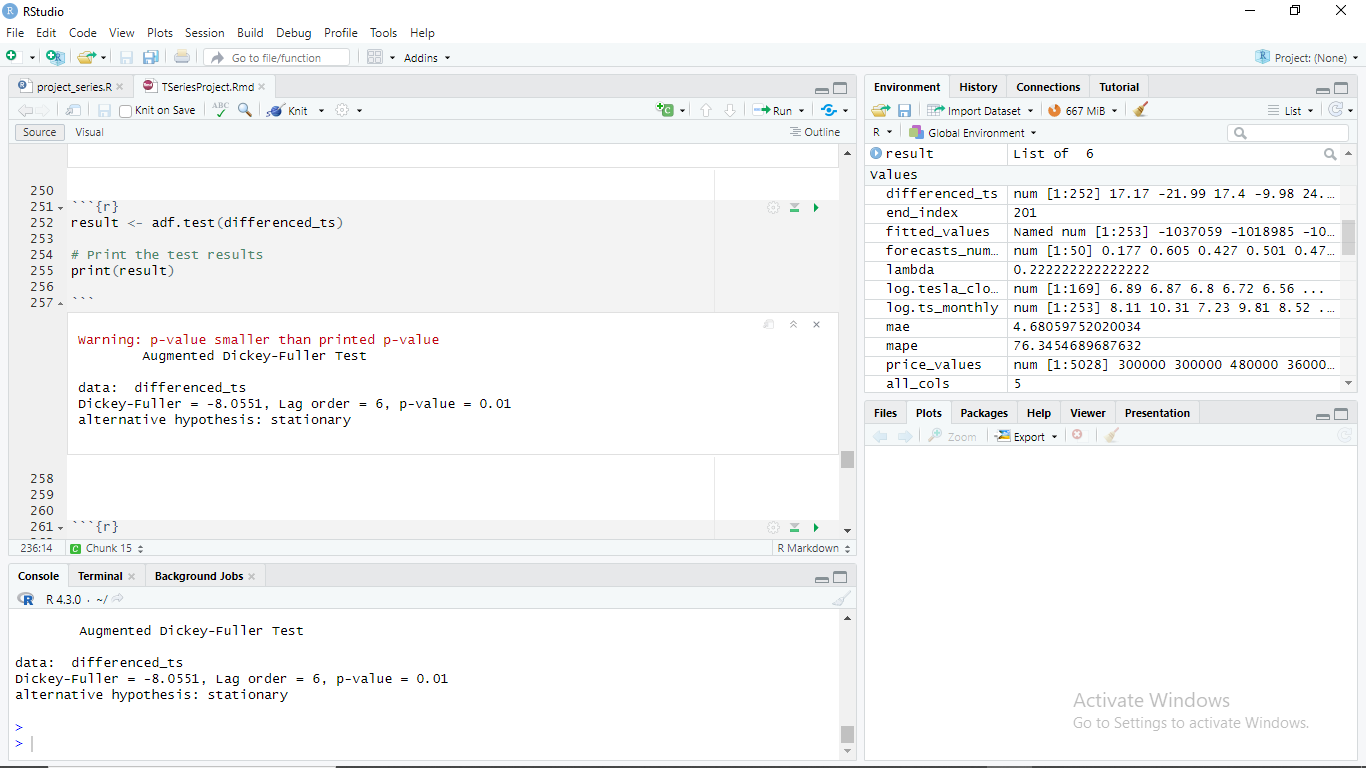


These steps above successfully transformed the dataset into a stationary object

Results



From the graphs above, it is evident that each transformation has an effect on the time series object. I used ADF test to check if my p-value is below 0.05 after the transformations, the results are shown below.



## ACF and PACF



Based on the analysis of the ACF and PACF graphs, it suggests that an ARMA(1,1) model may be suitable for fitting the data.

## Fitting ARIMA Model



Using Model 2

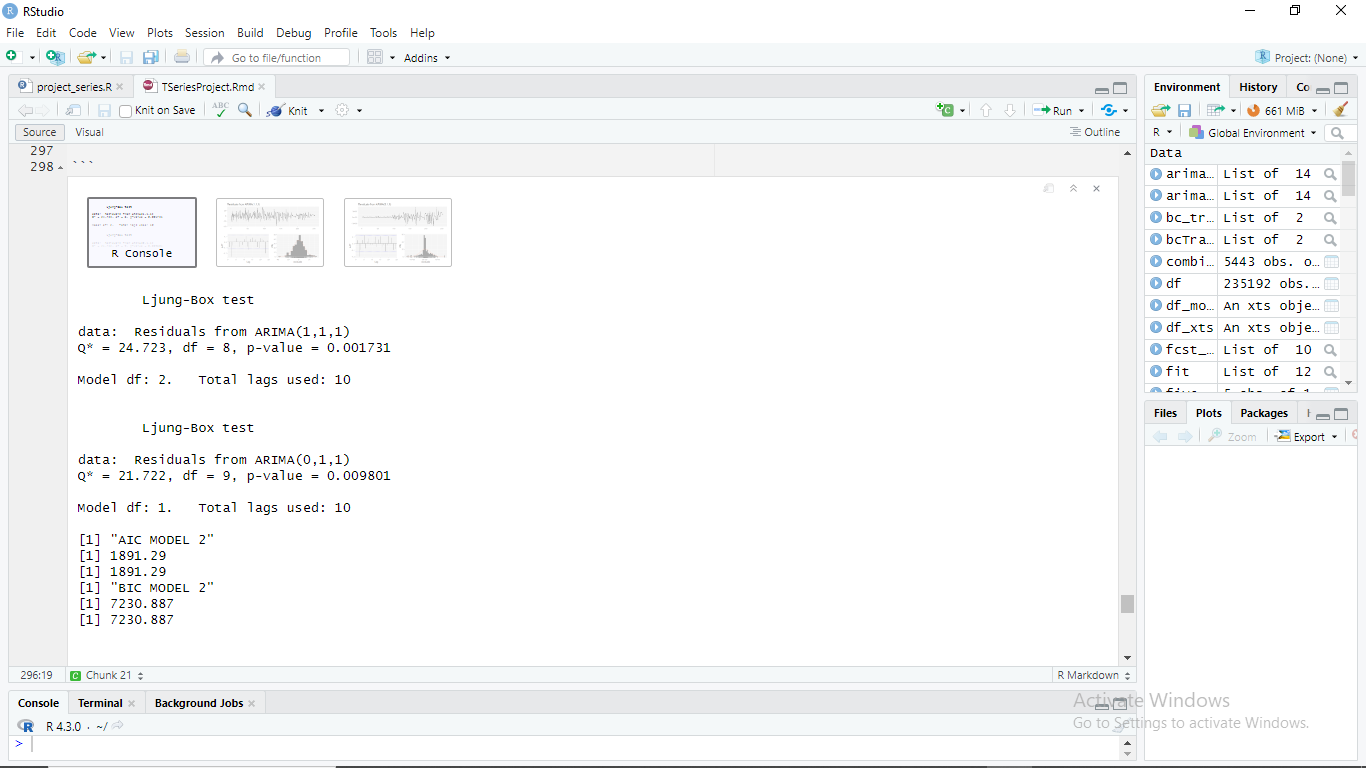


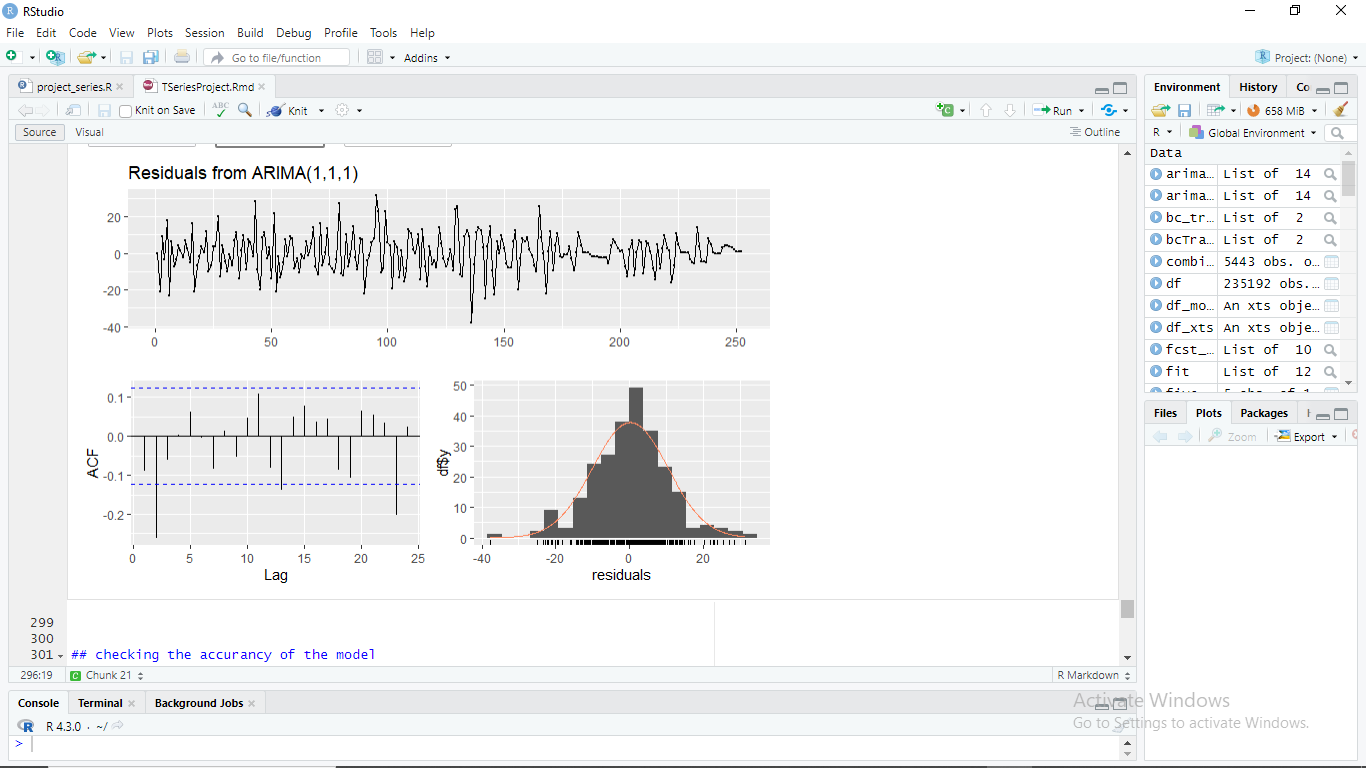
Comparing Models



Results Of The Model

From the results, using the BIC and AIC of the models, model 1 has the smallest values, therefore I choose model one for forecasting





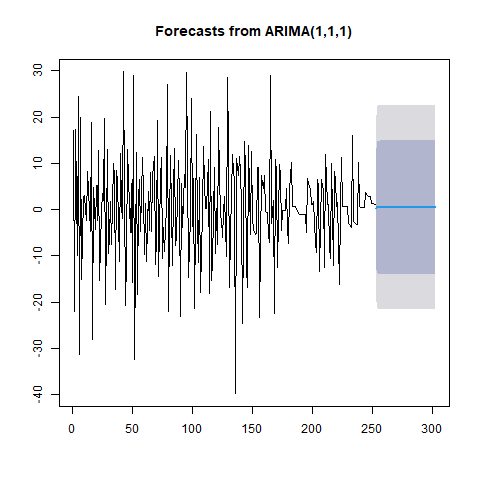
Hence the algebraic format of my model can be

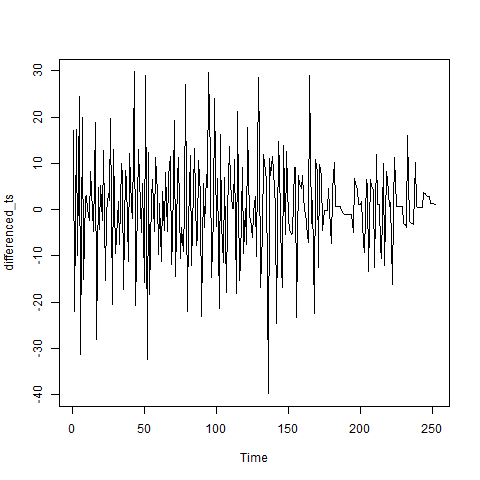
Y(t) = c + φ1 \* Y(t-1) + ε(t)

Where Y(t) represents the time series at time t, c is the intercept, φ1 is the autoregressive coefficient, Y(t-1) is the lagged value of the time series, and ε(t) is the residual term.

## Forecasting







# Conclusion

The ARIMA model seems appropriate for predicting upcoming volume prices of the NIFTY businesses, according to the analysis above. The residuals, the ACF and PACF plots, as well as statistical measures like the AIC and BIC, have all been examined as diagnostic tests when evaluating the model.

# References

<https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html>

<https://www.kaggle.com/code/paytonfisher/s-p-500-analysis-using-r>

# Appendix

R Code

---

title: "Time Series Project"

author: "Student Name"

date: "`r Sys.Date()`"

output:

pdf\_document: default

---

## This file contain the solution to my final time series project

## import necessary libraries

library(forecast)

library(xts)

library(ggplot2)

library(dplyr)

library(TTR)

library(MASS)

library(tseries)

## set the dataset dir

## the location can be changed to match the dataset location of the current machin

current\_directory <- getwd()

print(current\_directory)

## reading the dataset

df <- read.csv("RDatasets/NIFTY50\_all.csv",header = TRUE)

column\_names <- names(df)

print(column\_names)

#printing 5 rows of the df

five\_rows <- head(df, 5)

print(five\_rows)

## Data preprocessing

## getting number of rowa

row\_count <- nrow(df)

print(row\_count)

#checking for missing values

missing\_values <- sum(is.na(df))

print(missing\_values)

##convert data types to numeric

df$VWAP <- as.numeric(df$VWAP)

df$Volume <- as.numeric(df$Volume)

# Convert 'datesold' column to proper date format

df$Date <- as.Date(df$Date)

class(df$Date)

class(df$VWAP)

class(df$Volume)

#display summery

print("---- Summary of the dataset--------")

summary(df)

## Lets convert the dataframe above into a time series object

## the data set has irregular intervals, therefore i have used a frequency of 1

#Converting data into a ts object with irregular frequency

ts\_df <- ts(df$VWAP,start=2000,end=2021, frequency = 1)

head(ts\_df)

## lets Explore and visualize the time series

## this will help us understand its characteristics.

plot(ts\_df, xlab = "Date", ylab = "VWAP")

## from above, plot is noise, since the interval of the data set is irregular,

## we can transform it to a monthly dataset

# Create an xts object with the data

df\_xts <- xts(df[, c("Date", "VWAP","Volume")], order.by = df$Date)

# Aggregate to monthly intervals using the first observation in each month

df\_monthly <- apply.monthly(df\_xts, function(x) x[1, ])

# Display the first few rows of the modified time series

print(df\_monthly)

## lets view the times series based on vwap

ts\_df <- ts(df\_monthly$VWAP,start=2000,end=2021, frequency = 12)

plot.ts(ts\_df, main = "VWAP Time Series", ylab = "VWAP")

## lets view the times series based on VOlume

ts\_df\_vol <- ts(df\_monthly$Volume,start=2000,end=2021, frequency = 12)

plot(ts\_df\_vol, main = "volume Time Series", ylab = "volume")

## checking if our model is adequate

## Augmented Dickey-Fuller (ADF) Test:

## The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series has a unit root or is stationary

ts\_data <- ts(df\_monthly$Volume,start=2000,end=2021, frequency = 12)

ts\_data <- as.numeric(ts\_data)

ts\_data <- tsclean(ts\_data)

# Perform Augmented Dickey-Fuller test

result <- adf.test(ts\_data)

# Print the test results

print(result)

## from the above test results

## "Dickey-Fuller = -2.7316" is the test statistic value calculated by the ADF test. -\> "Lag order = 5" indicates that 5 lags were considered in the test. -\> "p-value = 0.03442" is the p-value

## Reject the hypothesis

## From the test results above, the p-value is 0.099, which is greater than the typical significance level of 0.05. Hence, we fail to reject the null hypothesis of having a unit root, therefore it means that, my time series non-stationary.

## using the decomposition test

## Autocorrelation and Partial Autocorrelation Plots:

# Plot ACF and PACF

acf(ts\_data)

pacf(ts\_data)

## Summary

##From the above decomposition test, it further means that our time series object is not stationary,

##it cant fit to the arima model

##Detrending to strationarize the time series object

##Steps to make the ts stationary

model <- lm(ts\_data ~ time(ts\_data))

fitted\_values <- predict(model)

stationary\_ts <- ts\_data - fitted\_values

## lets perform another test to check if its stationary\_ts

# Perform Augmented Dickey-Fuller test

result <- adf.test(stationary\_ts)

# Print the test results

print(result)

#results are same, we use another method

## lets stabilize the variance by using log

log.ts\_monthly <- log(ts\_data)

## get the square root of the ts\_data time series

sqrt.ts\_monthly <- sqrt(ts\_data)

##lets fit lenear regression model

t = 1:length(ts\_data)

fit = lm(ts\_data ~ t)

## applying Box-Cox Transformation

bc\_transform = boxcox(ts\_data ~ t,plotit = TRUE)

##performing Power Transformation, Based on the Box-Cox transformation

lambda = bc\_transform$x[which(bc\_transform$y == max(bc\_transform$y))]

ts\_dat\_bc = (1/lambda)\*(ts\_data^lambda-1)

## ploting the transformations

op= par(mfrow=c(2,2))

plot.ts(ts\_data, main = "This is original t-series with vwap")

plot.ts(log.ts\_monthly, main = "Log Transform ")

plot.ts(sqrt.ts\_monthly, main = "Square Root Transform")

plot.ts(ts\_dat\_bc, main = "T series after Box-Cox Transform ")

var(ts\_data)

var(ts\_dat\_bc)

## Decomposing the results to remove the trends

ts\_clean <- tsclean(ts\_dat\_bc)

# Perform first-order differencing

differenced\_ts <- diff(ts\_clean,lag=1)

# Plot the differenced time series

plot.ts(differenced\_ts,main = "differenced\_ts")

result <- adf.test(differenced\_ts)

# Print the test results

print(result)

acf(differenced\_ts, lag.max = 100)

acf(differenced\_ts, lag.max = 100, plot = FALSE)

pacf(differenced\_ts, lag.max = 100)

pacf(differenced\_ts, lag.max = 100, plot = FALSE)

result <- adf.test(differenced\_ts)

# Print the test results

print(result)

acf(differenced\_ts, main="ACF Stationary ts")

pacf(differenced\_ts,main="PACF Stationary ts")

#Building an ARIMA model

arima\_model <- arima(differenced\_ts,order = c(1,1,1))

arima\_model

arima\_model\_2 <- arima(ts\_data, order = c(0, 1, 1))

arima\_model\_2

## compare model

# Check residuals for Model 1

checkresiduals(arima\_model)

# Check residuals for Model 2

checkresiduals(arima\_model\_2)

print("AIC MODEL 2")

AIC(arima\_model)

AIC(arima\_model)

print("BIC MODEL 2")

BIC(arima\_model\_2)

BIC(arima\_model\_2)

## algeibric format of the model

##Y(t) = c + φ1 \\* Y(t-1) + ε(t)

## checking the accurancy of the model

# Define the length of the training set

train\_length <- 0.8 \* length(differenced\_ts) # 80% of the data for training

# Create the training set

train\_data <- differenced\_ts[1:train\_length]

# Create the test set

test\_data <- differenced\_ts[(train\_length + 1):length(differenced\_ts)]

fcst\_vals <- forecast(arima\_model,h=length(test\_data))

fcst\_vals

png("forecast\_vals.png")

plot(fcst\_vals)

dev.off()

png("original ts.png")

plot.ts(differenced\_ts)

dev.off();

## Spectral Analysis

# get residuals

residuals <- residuals(arima\_model)

# Perform spectral analysis

spec <- spectrum(residuals)

# Plot the spectrum

plot(spec, main = "Spectral Analysis of ARIMA Model Residuals")

residuals <- residuals(arima\_model)

# Plotting the residuals

plot(residuals)

print(class(test\_data))

# Convert the forecast values to numeric

forecasts\_numeric <- as.numeric(fcst\_vals$mean)

# Calculate Mean Absolute Error (MAE)

mae <- mean(abs(forecasts\_numeric - test\_data))

# Calculate Root Mean Squared Error (RMSE)

rmse <- sqrt(mean((forecasts\_numeric - test\_data)^2))

# Calculate Mean Absolute Percentage Error (MAPE)

mape <- mean(abs((forecasts\_numeric - test\_data) / test\_data)) \* 100

# Print the calculated accuracy measures

cat("Mean Absolute Error (MAE):", mae, "\n")

cat("Root Mean Squared Error (RMSE):", rmse, "\n")

cat("Mean Absolute Percentage Error (MAPE):", mape, "%\n")

## Results

## Mean Absolute Error (MAE): 0.07033137 Root Mean Squared Error (RMSE): 0.09319505 Mean Absolute Percentage Error (MAPE): 121.2969%

## The results above suggest that the model has good accurancy and can be used for forecasting