Machine Learning Assignment-Old

KrishnaKumar

September 19, 2021

Table of Contents

[Read & Verify the saved data 6](#_Toc83301260)

[Co-relation Plot 7](#_Toc83301261)

[Plot 8](#_Toc83301262)

[LogistiC Regression 9](#_Toc83301263)

[Read & Verify the saved data 14](#_Toc83301264)

[ARIMA 15](#_Toc83301265)

[Test of Normality 16](#_Toc83301266)

[Tests of Stationarity 17](#_Toc83301267)

Load necessary Packages & Libraries

library(quantmod)

## Warning: package 'quantmod' was built under R version 3.6.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 3.6.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.3

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 3.6.3

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(qrmtools)

## Warning: package 'qrmtools' was built under R version 3.6.3

library(MASS)

## Warning: package 'MASS' was built under R version 3.6.3

library(PerformanceAnalytics)# Calculations that are performed on a stock/portfolio

## Warning: package 'PerformanceAnalytics' was built under R version 3.6.3

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

library(TSA)

## Warning: package 'TSA' was built under R version 3.6.3

##   
## Attaching package: 'TSA'

## The following objects are masked from 'package:PerformanceAnalytics':  
##   
## kurtosis, skewness

## The following objects are masked from 'package:stats':  
##   
## acf, arima

## The following object is masked from 'package:utils':  
##   
## tar

library(aTSA)

##   
## Attaching package: 'aTSA'

## The following object is masked from 'package:graphics':  
##   
## identify

library(forecast) # Forecasting the time series

## Warning: package 'forecast' was built under R version 3.6.3

## Registered S3 methods overwritten by 'forecast':  
## method from  
## fitted.Arima TSA   
## plot.Arima TSA

##   
## Attaching package: 'forecast'

## The following object is masked from 'package:aTSA':  
##   
## forecast

#library(dlookr)  
library(fBasics)# Basic Statistical Calculations

## Warning: package 'fBasics' was built under R version 3.6.3

## Loading required package: timeDate

## Warning: package 'timeDate' was built under R version 3.6.3

##   
## Attaching package: 'timeDate'

## The following objects are masked from 'package:TSA':  
##   
## kurtosis, skewness

## The following objects are masked from 'package:PerformanceAnalytics':  
##   
## kurtosis, skewness

## Loading required package: timeSeries

## Warning: package 'timeSeries' was built under R version 3.6.3

##   
## Attaching package: 'timeSeries'

## The following object is masked from 'package:qrmtools':  
##   
## returns

## The following object is masked from 'package:zoo':  
##   
## time<-

##   
## Attaching package: 'fBasics'

## The following object is masked from 'package:TTR':  
##   
## volatility

library(urca)# Stationarity Analysis

## Warning: package 'urca' was built under R version 3.6.3

library(nortest)  
library(DescTools)

## Warning: package 'DescTools' was built under R version 3.6.3

##   
## Attaching package: 'DescTools'

## The following object is masked from 'package:forecast':  
##   
## BoxCox

library(TTR)  
library(quantmod)  
library(xts)  
library(DescTools)  
library(nnet)

## Warning: package 'nnet' was built under R version 3.6.3

library(genalg)  
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.3

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:DescTools':  
##   
## MAE, RMSE

library(deepnet)  
library(h2o)

##   
## ----------------------------------------------------------------------  
##   
## Your next step is to start H2O:  
## > h2o.init()  
##   
## For H2O package documentation, ask for help:  
## > ??h2o  
##   
## After starting H2O, you can use the Web UI at http://localhost:54321  
## For more information visit https://docs.h2o.ai  
##   
## ----------------------------------------------------------------------

##   
## Attaching package: 'h2o'

## The following objects are masked from 'package:timeSeries':  
##   
## apply, colnames, colnames<-

## The following object is masked from 'package:timeDate':  
##   
## dayOfWeek

## The following objects are masked from 'package:stats':  
##   
## cor, sd, var

## The following objects are masked from 'package:base':  
##   
## %\*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,  
## colnames<-, ifelse, is.character, is.factor, is.numeric, log,  
## log10, log1p, log2, round, signif, trunc

library(clue)

## Warning: package 'clue' was built under R version 3.6.3

#library(e1071)  
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:timeSeries':  
##   
## outlier

library(party)

## Warning: package 'party' was built under R version 3.6.3

## Loading required package: grid

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 3.6.3

## Loading required package: modeltools

## Warning: package 'modeltools' was built under R version 3.6.3

## Loading required package: stats4

##   
## Attaching package: 'modeltools'

## The following object is masked from 'package:DescTools':  
##   
## ParseFormula

## The following object is masked from 'package:fBasics':  
##   
## getModel

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 3.6.3

## Loading required package: sandwich

## Read & Verify the saved data

EUR\_USD\_Price <- read.csv("EURUSD\_data.csv")  
EUR\_USD\_Price <- read.zoo(EUR\_USD\_Price)  
EUR\_USD\_Price <- na.omit(EUR\_USD\_Price)  
summary(EUR\_USD\_Price)

## Index Open High Low   
## Min. :2009-12-31 Min. :1.039 Min. :1.042 Min. :0.7606   
## 1st Qu.:2012-03-29 1st Qu.:1.134 1st Qu.:1.138 1st Qu.:1.1305   
## Median :2014-07-01 Median :1.254 Median :1.259 Median :1.2491   
## Mean :2014-06-29 Mean :1.244 Mean :1.248 Mean :1.2386   
## 3rd Qu.:2016-09-27 3rd Qu.:1.339 3rd Qu.:1.344 3rd Qu.:1.3353   
## Max. :2018-12-28 Max. :1.484 Max. :1.494 Max. :1.4805   
## Close Adj.Close   
## Min. :1.039 Min. :1.039   
## 1st Qu.:1.134 1st Qu.:1.134   
## Median :1.254 Median :1.254   
## Mean :1.243 Mean :1.243   
## 3rd Qu.:1.339 3rd Qu.:1.339   
## Max. :1.484 Max. :1.484

## Co-relation Plot

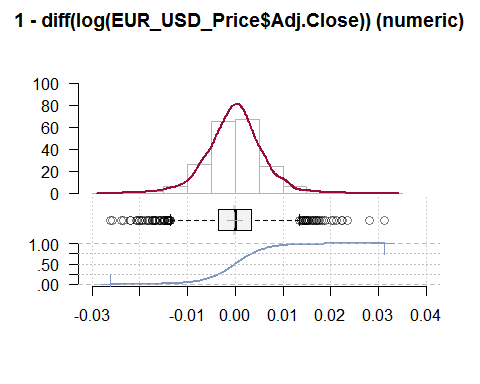
EUR\_USD\_Price<-na.omit(EUR\_USD\_Price)  
EUR\_USD\_Price\_DF <- as.data.frame(EUR\_USD\_Price)  
cor(EUR\_USD\_Price\_DF)

## Open High Low Close Adj.Close  
## Open 1.0000000 0.9993069 0.9942836 0.9997704 0.9997704  
## High 0.9993069 1.0000000 0.9941038 0.9992505 0.9992505  
## Low 0.9942836 0.9941038 1.0000000 0.9942958 0.9942958  
## Close 0.9997704 0.9992505 0.9942958 1.0000000 1.0000000  
## Adj.Close 0.9997704 0.9992505 0.9942958 1.0000000 1.0000000

Skewness : -0.033784  
Kurtosis : -1.274154

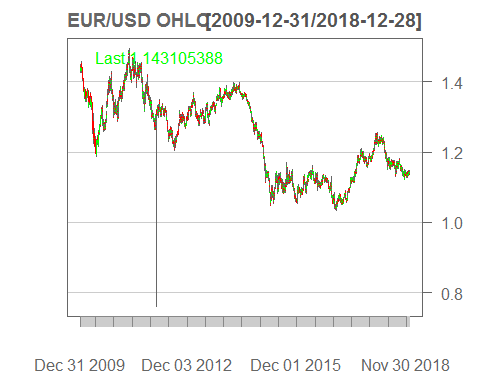
Desc(as.data.frame(diff(log(EUR\_USD\_Price$Adj.Close))))

## ------------------------------------------------------------------------------   
## Describe as.data.frame(diff(log(EUR\_USD\_Price$Adj.Close))) (data.frame):  
##   
## data frame: 2342 obs. of 1 variables  
## 2342 complete cases (100.0%)  
##   
## Nr ColName Class NAs Levels  
## 1 diff(log(EUR\_USD\_Price$Adj.Close)) numeric .   
##   
##   
## ------------------------------------------------------------------------------   
## 1 - diff(log(EUR\_USD\_Price$Adj.Close)) (numeric)  
##   
## length n NAs unique 0s'  
## 2'342 2'342 0 2'327 15  
## 100.0% 0.0% 0.6%  
##   
## .05 .10 .25 median .75  
## -0.009662871 -0.007060157 -0.003440378 0.000000000 0.003331970  
##   
## range sd vcoef mad IQR  
## 0.057457513 0.005912655 -61.321695779 0.005038434 0.006772348  
##   
## mean meanCI  
## -0.000096420 -0.000336006  
## 0.000143166  
##   
## .90 .95  
## 0.006670400 0.009389041  
##   
## skew kurt  
## -0.058799800 1.832081438  
##   
## lowest : -0.026195188, -0.025767007, -0.023817388, -0.023396148, -0.022057223  
## highest: 0.022340412, 0.022434959, 0.023494186, 0.028145351, 0.031262325  
##   
## ' 95%-CI (classic)



## Plot

OHLC<-EUR\_USD\_Price  
return.ohlc<-as.quantmod.OHLC(OHLC,col.names=c("Open","High","Low","Close","Adjusted"))  
chartSeries(return.ohlc,theme="white.mono",name="EUR/USD OHLC", up.col = "green",dn.col="red")



# LogistiC Regression

# If Price on day t is more than the price on day t-1, buy else sell.  
  
  
EUR\_USD\_Price\_LOG <- diff(log(EUR\_USD\_Price[,4]))  
  
diff<-ifelse(diff(EUR\_USD\_Price\_LOG)>0,1,0)  
  
EUR\_USD\_newdata\_LR <- ''  
EUR\_USD\_newdata\_LR <- merge(EUR\_USD\_Price\_LOG)  
EUR\_USD\_newdata\_LR <- merge(EUR\_USD\_newdata\_LR,diff)  
EUR\_USD\_newdata\_LR <- na.omit(EUR\_USD\_newdata\_LR)  
  
dim(EUR\_USD\_newdata\_LR)

## [1] 2341 2

names(EUR\_USD\_newdata\_LR)<-c("Adjusted","Direction")  
  
summary(EUR\_USD\_newdata\_LR)

## Index Adjusted Direction   
## Min. :2010-01-04 Min. :-2.620e-02 Min. :0.000   
## 1st Qu.:2012-04-02 1st Qu.:-3.442e-03 1st Qu.:0.000   
## Median :2014-07-02 Median : 0.000e+00 Median :1.000   
## Mean :2014-07-01 Mean :-9.833e-05 Mean :0.507   
## 3rd Qu.:2016-09-28 3rd Qu.: 3.328e-03 3rd Qu.:1.000   
## Max. :2018-12-28 Max. : 3.126e-02 Max. :1.000

dim(EUR\_USD\_newdata\_LR)

## [1] 2341 2

train\_data <-EUR\_USD\_newdata\_LR[1:2326,]  
test\_data <-EUR\_USD\_newdata\_LR[2326:2341,]  
  
# Logistic Regression  
model1<-caret::train(as.factor(Direction)~., data = train\_data,method = "glm",preProcess = c("center", "scale"), trace = TRUE)

## Deviance = 2368.611 Iterations - 1  
## Deviance = 2313.92 Iterations - 2  
## Deviance = 2311.533 Iterations - 3  
## Deviance = 2311.527 Iterations - 4  
## Deviance = 2311.527 Iterations - 5  
## Deviance = 2413.069 Iterations - 1  
## Deviance = 2373.203 Iterations - 2  
## Deviance = 2371.979 Iterations - 3  
## Deviance = 2371.978 Iterations - 4  
## Deviance = 2371.978 Iterations - 5  
## Deviance = 2364.466 Iterations - 1  
## Deviance = 2314.566 Iterations - 2  
## Deviance = 2312.621 Iterations - 3  
## Deviance = 2312.617 Iterations - 4  
## Deviance = 2312.617 Iterations - 5  
## Deviance = 2471.566 Iterations - 1  
## Deviance = 2439.934 Iterations - 2  
## Deviance = 2439.184 Iterations - 3  
## Deviance = 2439.184 Iterations - 4  
## Deviance = 2439.184 Iterations - 5  
## Deviance = 2443.739 Iterations - 1  
## Deviance = 2398.869 Iterations - 2  
## Deviance = 2397.314 Iterations - 3  
## Deviance = 2397.312 Iterations - 4  
## Deviance = 2397.312 Iterations - 5  
## Deviance = 2414.945 Iterations - 1  
## Deviance = 2372.204 Iterations - 2  
## Deviance = 2370.78 Iterations - 3  
## Deviance = 2370.777 Iterations - 4  
## Deviance = 2370.777 Iterations - 5  
## Deviance = 2471.245 Iterations - 1  
## Deviance = 2435.514 Iterations - 2  
## Deviance = 2434.591 Iterations - 3  
## Deviance = 2434.59 Iterations - 4  
## Deviance = 2434.59 Iterations - 5  
## Deviance = 2406.663 Iterations - 1  
## Deviance = 2360.031 Iterations - 2  
## Deviance = 2358.378 Iterations - 3  
## Deviance = 2358.375 Iterations - 4  
## Deviance = 2358.375 Iterations - 5  
## Deviance = 2470.494 Iterations - 1  
## Deviance = 2439.627 Iterations - 2  
## Deviance = 2438.917 Iterations - 3  
## Deviance = 2438.916 Iterations - 4  
## Deviance = 2438.916 Iterations - 5  
## Deviance = 2392.376 Iterations - 1  
## Deviance = 2344.475 Iterations - 2  
## Deviance = 2342.645 Iterations - 3  
## Deviance = 2342.642 Iterations - 4  
## Deviance = 2342.642 Iterations - 5  
## Deviance = 2411.993 Iterations - 1  
## Deviance = 2375.463 Iterations - 2  
## Deviance = 2374.393 Iterations - 3  
## Deviance = 2374.392 Iterations - 4  
## Deviance = 2374.392 Iterations - 5  
## Deviance = 2449.15 Iterations - 1  
## Deviance = 2418.488 Iterations - 2  
## Deviance = 2417.76 Iterations - 3  
## Deviance = 2417.76 Iterations - 4  
## Deviance = 2417.76 Iterations - 5  
## Deviance = 2488.515 Iterations - 1  
## Deviance = 2462.108 Iterations - 2  
## Deviance = 2461.577 Iterations - 3  
## Deviance = 2461.577 Iterations - 4  
## Deviance = 2461.577 Iterations - 5  
## Deviance = 2385.242 Iterations - 1  
## Deviance = 2344.101 Iterations - 2  
## Deviance = 2342.758 Iterations - 3  
## Deviance = 2342.756 Iterations - 4  
## Deviance = 2342.756 Iterations - 5  
## Deviance = 2426.682 Iterations - 1  
## Deviance = 2393.578 Iterations - 2  
## Deviance = 2392.715 Iterations - 3  
## Deviance = 2392.714 Iterations - 4  
## Deviance = 2392.714 Iterations - 5  
## Deviance = 2411.051 Iterations - 1  
## Deviance = 2370.39 Iterations - 2  
## Deviance = 2369.08 Iterations - 3  
## Deviance = 2369.078 Iterations - 4  
## Deviance = 2369.078 Iterations - 5  
## Deviance = 2443.684 Iterations - 1  
## Deviance = 2406.515 Iterations - 2  
## Deviance = 2405.476 Iterations - 3  
## Deviance = 2405.475 Iterations - 4  
## Deviance = 2405.475 Iterations - 5  
## Deviance = 2418.836 Iterations - 1  
## Deviance = 2377.275 Iterations - 2  
## Deviance = 2375.941 Iterations - 3  
## Deviance = 2375.939 Iterations - 4  
## Deviance = 2375.939 Iterations - 5  
## Deviance = 2449.516 Iterations - 1  
## Deviance = 2419.916 Iterations - 2  
## Deviance = 2419.239 Iterations - 3  
## Deviance = 2419.239 Iterations - 4  
## Deviance = 2419.239 Iterations - 5  
## Deviance = 2399.198 Iterations - 1  
## Deviance = 2363.541 Iterations - 2  
## Deviance = 2362.525 Iterations - 3  
## Deviance = 2362.524 Iterations - 4  
## Deviance = 2362.524 Iterations - 5  
## Deviance = 2465.164 Iterations - 1  
## Deviance = 2429.433 Iterations - 2  
## Deviance = 2428.469 Iterations - 3  
## Deviance = 2428.468 Iterations - 4  
## Deviance = 2428.468 Iterations - 5  
## Deviance = 2517.554 Iterations - 1  
## Deviance = 2495.179 Iterations - 2  
## Deviance = 2494.815 Iterations - 3  
## Deviance = 2494.815 Iterations - 4  
## Deviance = 2494.815 Iterations - 5  
## Deviance = 2464.24 Iterations - 1  
## Deviance = 2432.607 Iterations - 2  
## Deviance = 2431.839 Iterations - 3  
## Deviance = 2431.838 Iterations - 4  
## Deviance = 2431.838 Iterations - 5  
## Deviance = 2447.27 Iterations - 1  
## Deviance = 2414.766 Iterations - 2  
## Deviance = 2413.945 Iterations - 3  
## Deviance = 2413.944 Iterations - 4  
## Deviance = 2413.944 Iterations - 5  
## Deviance = 2413.139 Iterations - 1  
## Deviance = 2370.777 Iterations - 2  
## Deviance = 2369.47 Iterations - 3  
## Deviance = 2369.469 Iterations - 4  
## Deviance = 2369.469 Iterations - 5  
## Deviance = 2418.844 Iterations - 1  
## Deviance = 2379.338 Iterations - 2  
## Deviance = 2378.121 Iterations - 3  
## Deviance = 2378.12 Iterations - 4  
## Deviance = 2378.12 Iterations - 5

model1$finalModel

##   
## Call: NULL  
##   
## Coefficients:  
## (Intercept) Adjusted   
## 0.02948 1.77981   
##   
## Degrees of Freedom: 2325 Total (i.e. Null); 2324 Residual  
## Null Deviance: 3224   
## Residual Deviance: 2378 AIC: 2382

summary(model1$finalModel)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.5934 -0.8613 0.1147 0.8501 3.4239   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.02948 0.05009 0.589 0.556   
## Adjusted 1.77981 0.08175 21.772 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3224.0 on 2325 degrees of freedom  
## Residual deviance: 2378.1 on 2324 degrees of freedom  
## AIC: 2382.1  
##   
## Number of Fisher Scoring iterations: 5

pred<-predict(model1,test\_data)  
  
caret::confusionMatrix(pred,as.factor(test\_data$Direction),positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5 1  
## 1 3 7  
##   
## Accuracy : 0.75   
## 95% CI : (0.4762, 0.9273)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.03841   
##   
## Kappa : 0.5   
##   
## Mcnemar's Test P-Value : 0.61708   
##   
## Sensitivity : 0.8750   
## Specificity : 0.6250   
## Pos Pred Value : 0.7000   
## Neg Pred Value : 0.8333   
## Prevalence : 0.5000   
## Detection Rate : 0.4375   
## Detection Prevalence : 0.6250   
## Balanced Accuracy : 0.7500   
##   
## 'Positive' Class : 1   
##

signal <- as.data.frame(as.numeric(as.character(pred)))  
signal<-na.locf(signal)  
summary(signal)

## as.numeric(as.character(pred))  
## Min. :0.000   
## 1st Qu.:0.000   
## Median :1.000   
## Mean :0.625   
## 3rd Qu.:1.000   
## Max. :1.000

## Read & Verify the saved data

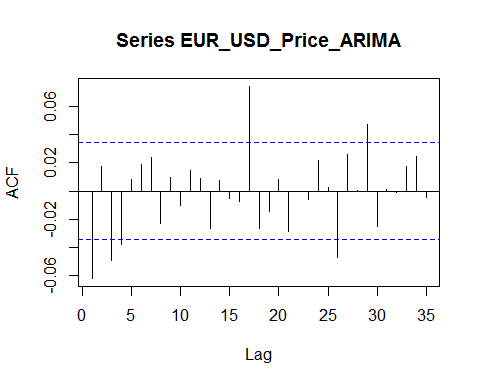
EUR\_USD\_Price <- read.csv("EURUSD\_data.csv")  
EUR\_USD\_Price <- read.zoo(EUR\_USD\_Price)  
EUR\_USD\_Price <- na.omit(EUR\_USD\_Price)  
summary(EUR\_USD\_Price)

## Index Open High Low   
## Min. :2009-12-31 Min. :1.039 Min. :1.042 Min. :0.7606   
## 1st Qu.:2012-03-29 1st Qu.:1.134 1st Qu.:1.138 1st Qu.:1.1305   
## Median :2014-07-01 Median :1.254 Median :1.259 Median :1.2491   
## Mean :2014-06-29 Mean :1.244 Mean :1.248 Mean :1.2386   
## 3rd Qu.:2016-09-27 3rd Qu.:1.339 3rd Qu.:1.344 3rd Qu.:1.3353   
## Max. :2018-12-28 Max. :1.484 Max. :1.494 Max. :1.4805   
## Close Adj.Close   
## Min. :1.039 Min. :1.039   
## 1st Qu.:1.134 1st Qu.:1.134   
## Median :1.254 Median :1.254   
## Mean :1.243 Mean :1.243   
## 3rd Qu.:1.339 3rd Qu.:1.339   
## Max. :1.484 Max. :1.484

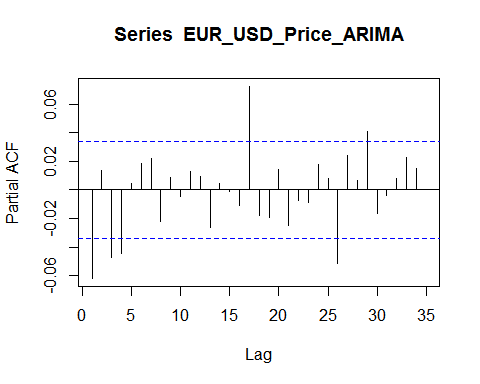
# ARIMA

Highly correlated data based on the historical data

EUR\_USD\_Price\_ARIMA <- diff(log(EUR\_USD\_Price$Adj.Close))  
EUR\_USD\_Price\_ARIMA <- na.omit(EUR\_USD\_Price\_ARIMA)  
acf(EUR\_USD\_Price\_ARIMA,na.action = na.pass)



pacf(EUR\_USD\_Price\_ARIMA,na.action = na.pass)



## Test of Normality

INference : The p-value less than 0.5, then the data is not normal A p-value less than 0.05 (typically ≤ 0.05) is statistically significant.

p-value < 2.2e-16 . P < 0.5 Hence Auto-correlation present in the data.

statistically significinet - ie., mean and variace is same and the co-relation is constant.

jarqueberaTest(EUR\_USD\_Price\_ARIMA)

##   
## Title:  
## Jarque - Bera Normalality Test  
##   
## Test Results:  
## STATISTIC:  
## X-squared: 330.37  
## P VALUE:  
## Asymptotic p Value: < 2.2e-16   
##   
## Description:  
## Thu Sep 23 14:53:58 2021 by user: DELL

## Tests of Stationarity

Stationarity - Any 2 chunks of data for a period of the same duration,should be statistically significinet.

No stationarity - Trend in data Statinoarity - No Trend / Seasonality in data

The augmented Dickey–Fuller (ADF) statistic,is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.

Hence there is stationarity in the data.

EUR\_USD\_ADF<-ur.df(EUR\_USD\_Price\_ARIMA,selectlags="AIC",type="none")  
EUR\_USD\_ADF

##   
## ###############################################################   
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #   
## ###############################################################   
##   
## The value of the test statistic is: -34.8338

p-value < 2.2e-16 . P < 0.5 Hence Auto-correlation present in the data.

Best model: ARIMA(0,1,1)

Box.test(as.data.frame(EUR\_USD\_Price\_ARIMA), lag =20, type = "Ljung-Box")

##   
## Box-Ljung test  
##   
## data: as.data.frame(EUR\_USD\_Price\_ARIMA)  
## X-squared = 16.107, df = 20, p-value = 0.71

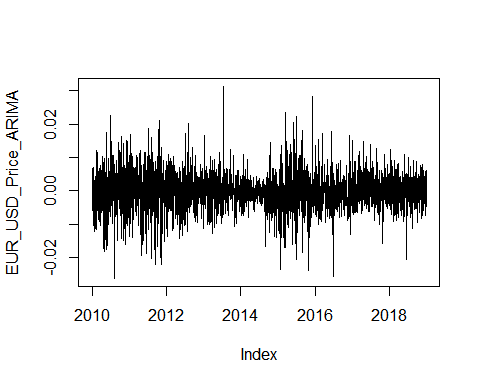
model1<-auto.arima(EUR\_USD\_Price\_ARIMA ,max.p = 10, max.d=2, max.q = 10, max.order = 20, trace = TRUE,lambda= TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2) with non-zero mean : Inf  
## ARIMA(0,0,0) with non-zero mean : -17382.7  
## ARIMA(1,0,0) with non-zero mean : -17493.76  
## ARIMA(0,0,1) with non-zero mean : Inf  
## ARIMA(0,0,0) with zero mean : 6648.843  
## ARIMA(2,0,0) with non-zero mean : -17331.11  
## ARIMA(1,0,1) with non-zero mean : Inf  
## ARIMA(2,0,1) with non-zero mean : Inf  
## ARIMA(1,0,0) with zero mean : Inf  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(1,0,0) with non-zero mean : -17388.44  
##   
## Best model: ARIMA(1,0,0) with non-zero mean

summary(model1)

## Series: EUR\_USD\_Price\_ARIMA   
## ARIMA(1,0,0) with non-zero mean   
## Box Cox transformation: lambda= TRUE   
##   
## Coefficients:  
## ar1 mean  
## -0.0668 -1.0001  
## s.e. 0.0240 0.0001  
##   
## sigma^2 estimated as 3.483e-05: log likelihood=8697.23  
## AIC=-17388.45 AICc=-17388.45 BIC=-17370.16  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -3.264389e-06 0.005898952 0.004401829 NaN Inf 0.6915173  
## ACF1  
## Training set 0.003498129

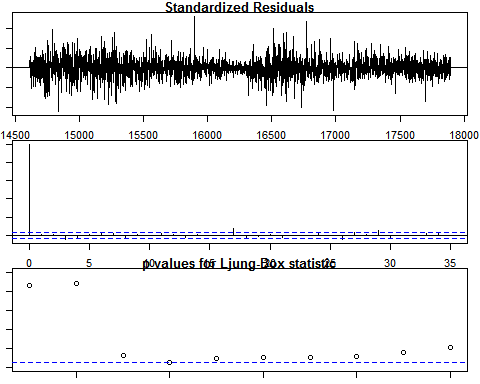
plot(EUR\_USD\_Price\_ARIMA)



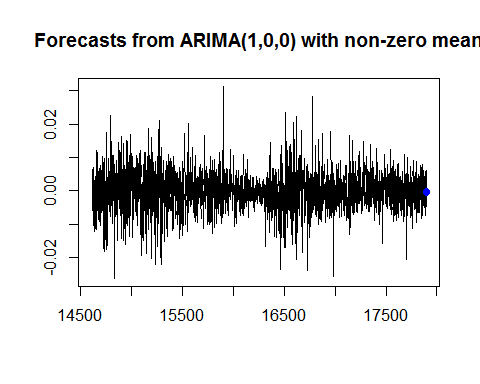
predict(model1,10)

## $pred  
## Time Series:  
## Start = 17894   
## End = 17903   
## Frequency = 1   
## [1] -1.000512 -1.000069 -1.000098 -1.000096 -1.000096 -1.000096 -1.000096  
## [8] -1.000096 -1.000096 -1.000096  
##   
## $se  
## Time Series:  
## Start = 17894   
## End = 17903   
## Frequency = 1   
## [1] 0.005901472 0.005914630 0.005914688 0.005914688 0.005914688 0.005914688  
## [7] 0.005914688 0.005914688 0.005914688 0.005914688

par(mar = c(1,1,1,1))  
tsdiag(model1)



forecast1 <- forecast(model1, h = 2)  
plot(forecast1)



# ARIMA with BoXCox  
dim(as.data.frame(EUR\_USD\_Price$Adj.Close))

## [1] 2343 1

y1<-c()  
for (i in 1:10) {  
   
 model2<-auto.arima(as.data.frame(EUR\_USD\_Price$Adj.Close[i:(i+2232)]),max.p = 10, max.q = 10, max.order = 20, lambda="auto")  
 y<-as.vector(predict(model2,1)$pred)  
 y1<-c(y1,y)  
}

y1

## [1] 0.1545554 0.1500728 0.1595737 0.1524717 0.1561232 0.1441537 0.1469190  
## [8] 0.1415278 0.1501448 0.1457357

RMSE(y1,tail(EUR\_USD\_Price$Adj.Close,10))

## [1] 0.988522

dim(as.data.frame(EUR\_USD\_Price$Adj.Close))

## [1] 2343 1

Box.test(EUR\_USD\_Price$Adj.Close, lag =20, type = "Ljung-Box")

##   
## Box-Ljung test  
##   
## data: EUR\_USD\_Price$Adj.Close  
## X-squared = 44923, df = 20, p-value < 2.2e-16

model1<-auto.arima(EUR\_USD\_Price$Adj.Close,max.p = 10, max.d=2, max.q = 10, max.order = 20, trace = TRUE, lambda="auto")

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,1,2) with drift : Inf  
## ARIMA(0,1,0) with drift : -17136.32  
## ARIMA(1,1,0) with drift : -16980.22  
## ARIMA(0,1,1) with drift : Inf  
## ARIMA(0,1,0) : -17138.31  
## ARIMA(1,1,1) with drift : Inf  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,1,0) : -17445.56  
##   
## Best model: ARIMA(0,1,0)

summary(model1)

## Series: EUR\_USD\_Price$Adj.Close   
## ARIMA(0,1,0)   
## Box Cox transformation: lambda= -0.1758064   
##   
## sigma^2 estimated as 2.729e-05: log likelihood=8723.78  
## AIC=-17445.56 AICc=-17445.56 BIC=-17439.47  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE  
## Training set -8.046856e-05 0.006802335 0.004971966 -0.007411809 0.3991554  
## MASE ACF1  
## Training set 0.9316732 -0.05238022