ARIMA USJap CurrencyRate

Jane

27/04/2021

Forcasting Exchange Rate Using GARCH Model for US Dollar and Japenese Yen

Reading Canadian and Japanes Currency into r

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
USJapCurrencyARIMA <-  read.csv ("USDJPY_Candlestick_1_D_BID_01.01.2000-31.12.2020.csv")%>%
  select('GMT.TIME', CLOSE)%>%
  rename(Date = ('GMT.TIME'), RateUSJapan = ("CLOSE"))


head(USJapCurrencyARIMA)
```

```
##         Date RateUSJapan
## 1 2000-01-03      101.48
## 2 2000-01-04      103.25
## 3 2000-01-05      104.30
## 4 2000-01-06      105.26
## 5 2000-01-07      105.31
## 6 2000-01-10      105.06
```

Conversion of Gmt time to date format

```r
library(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
USJapCurrencyARIMA$Date <- lubridate::ymd(USJapCurrencyARIMA$Date)
head(USJapCurrencyARIMA)
```

```
##         Date RateUSJapan
## 1 2000-01-03      101.48
## 2 2000-01-04      103.25
## 3 2000-01-05      104.30
## 4 2000-01-06      105.26
## 5 2000-01-07      105.31
## 6 2000-01-10      105.06
```

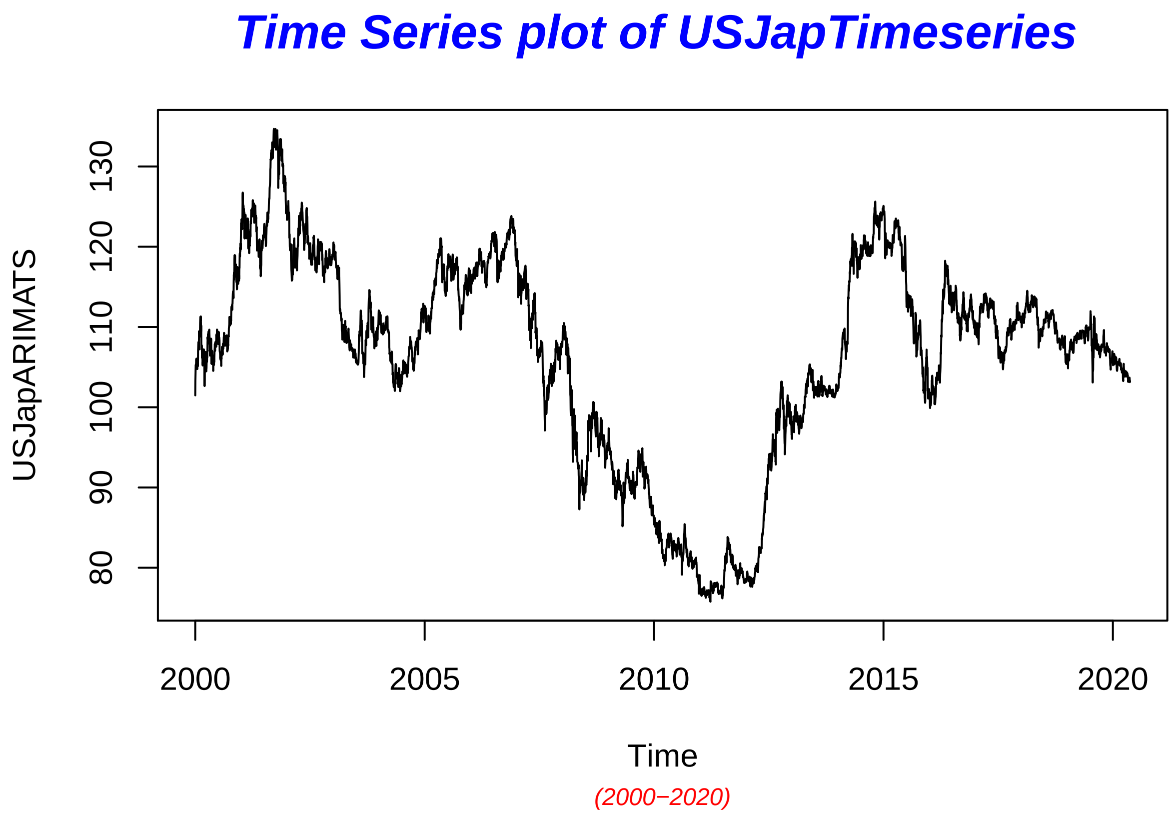
##Checking for obvious errors

```r
#Checking for obvious errors
which(is.na(USJapCurrencyARIMA))
```

```
## integer(0)
```

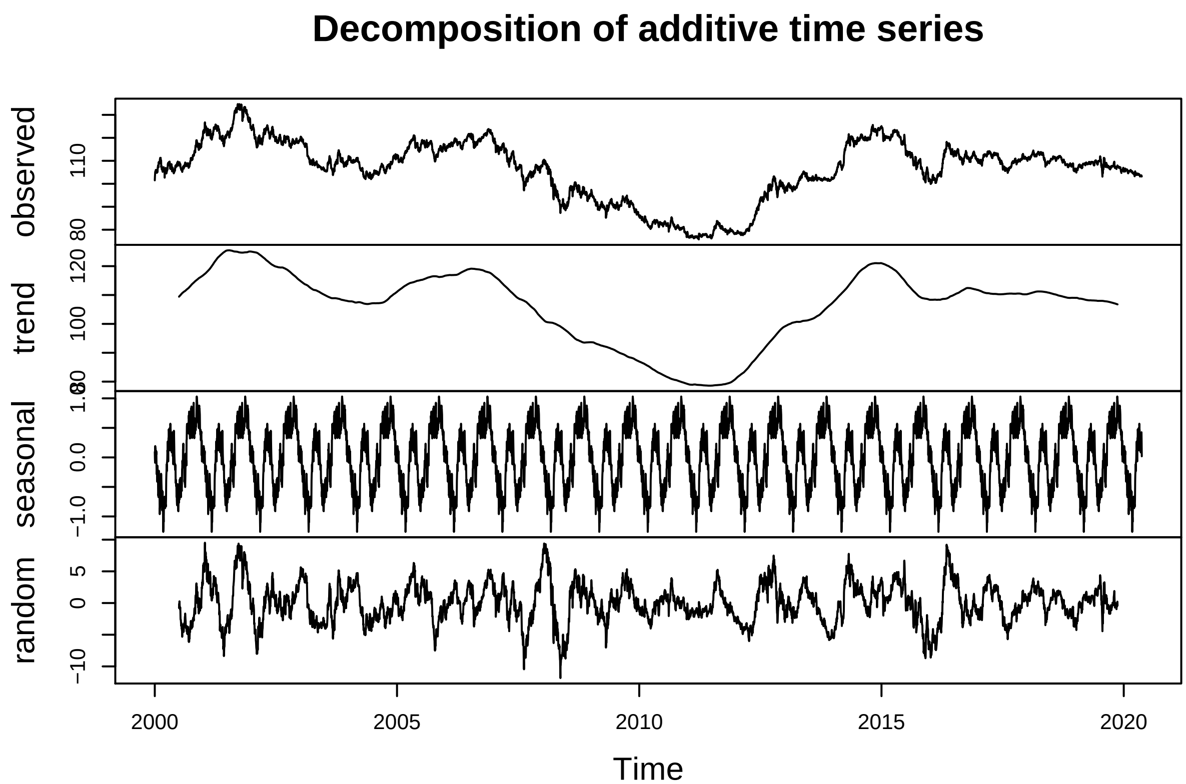
##Converting the data set into time series object

```r
#Converting the data set into time series object
USJapARIMATS<- ts(as.vector(USJapCurrencyARIMA$Rate),  frequency = 314, start= c(2000,01,03))
plot.ts(USJapARIMATS)
title("Time Series plot of USJapTimeseries ", sub = "(2000-2020)",
      cex.main = 1.5,   font.main= 4, col.main= "blue",
      cex.sub = 0.75, font.sub = 3, col.sub = "red")
```



Finding the component of the Time Series

```
ComponentUSJapCurrency <- decompose(USJapARIMATS)
plot(ComponentUSJapCurrency)
```

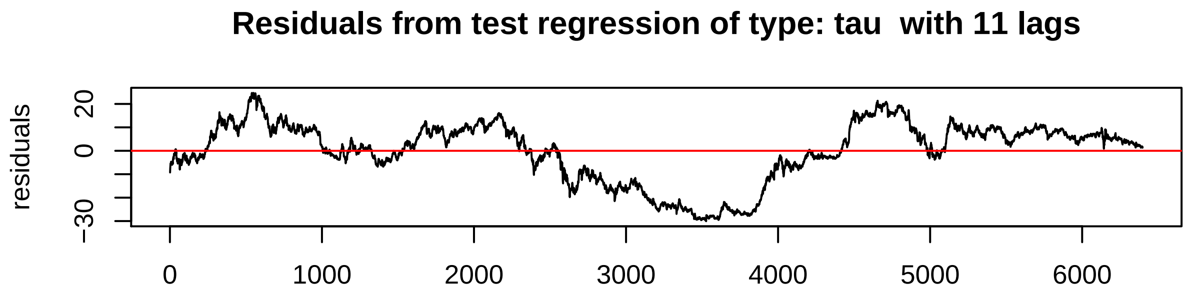


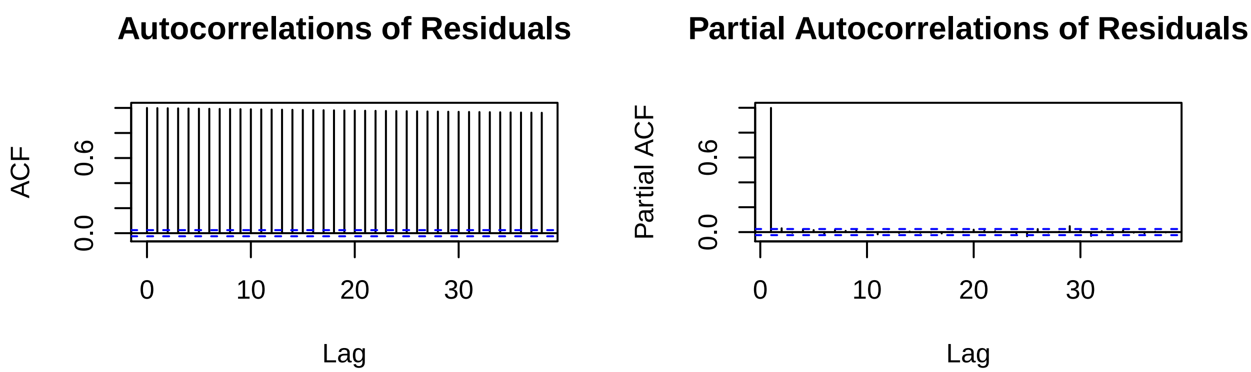
To To achieve stationarity by differencing the data – compute the differences between consecutive observations

```
library("fUnitRoots")
```

```
## Warning: package 'fUnitRoots' was built under R version 4.0.5

## Loading required package: timeDate

## Warning: package 'timeDate' was built under R version 4.0.4

## Loading required package: timeSeries

## Warning: package 'timeSeries' was built under R version 4.0.5

## Loading required package: fBasics

## Warning: package 'fBasics' was built under R version 4.0.5
```

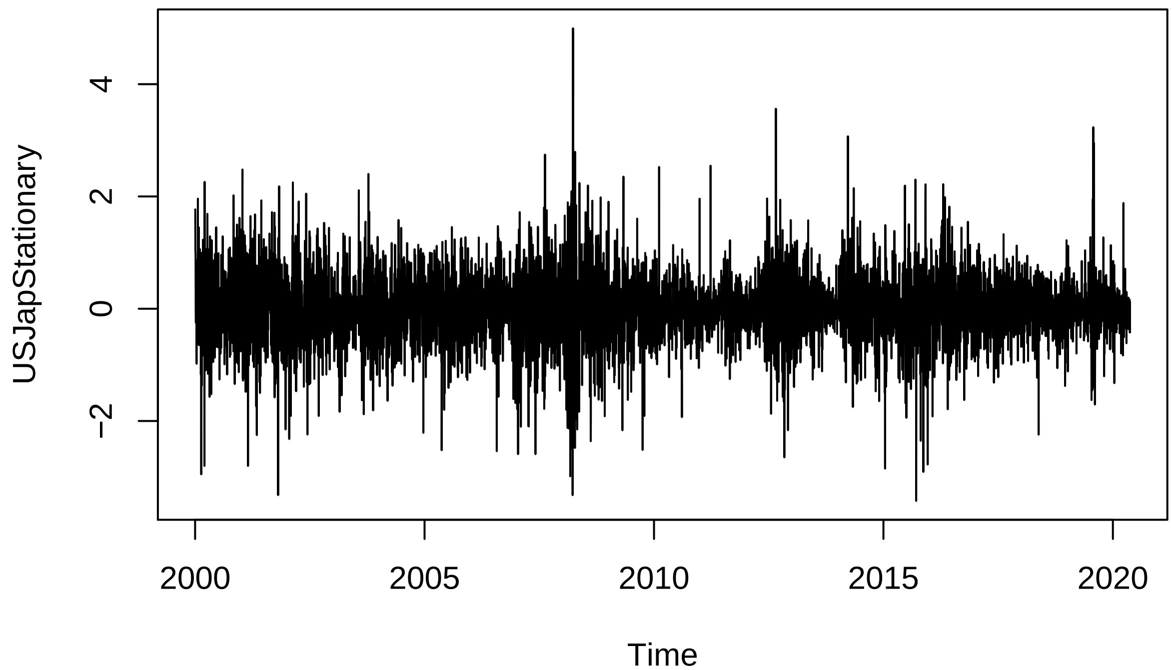
```r
urkpssTest(USJapARIMATS, type = c("tau"), lags = c("short"),use.lag = NULL, doplot = TRUE)
```





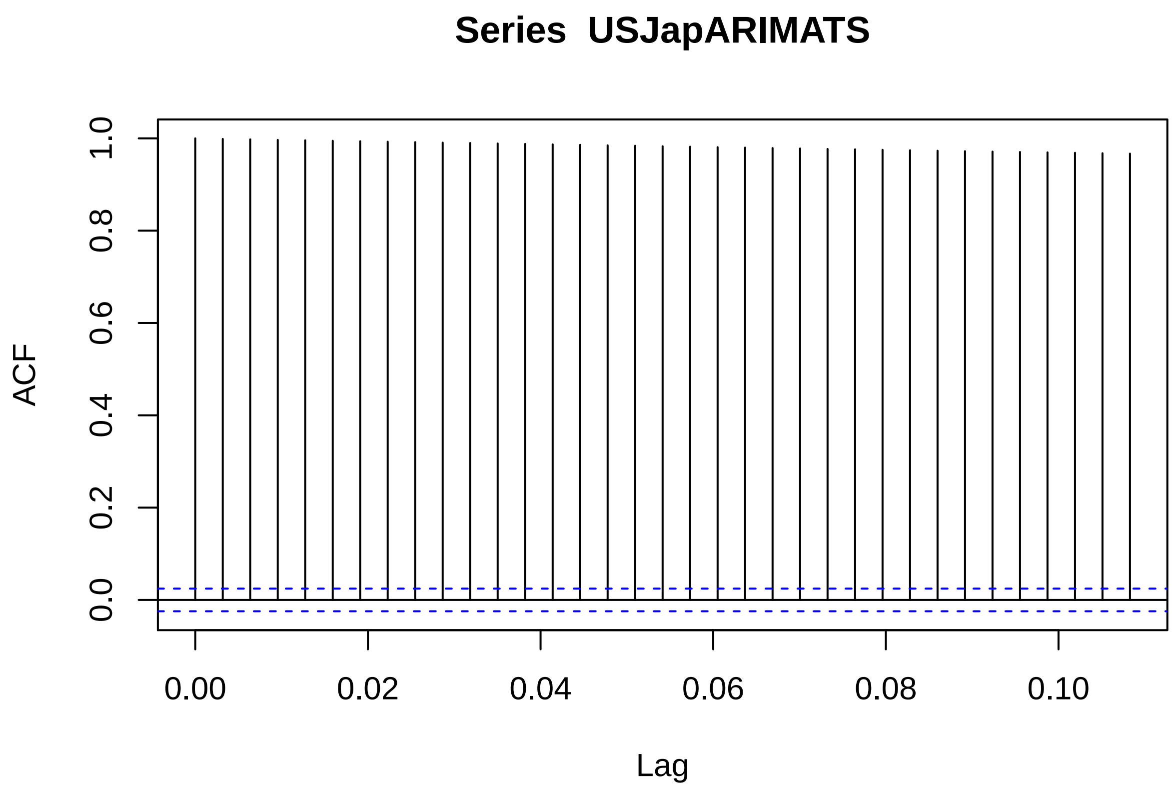

```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Tue May 04 00:53:31 2021 by user: janeo
```

```r
USJapStationary= diff(USJapARIMATS, differences=1)
plot(USJapStationary)
```

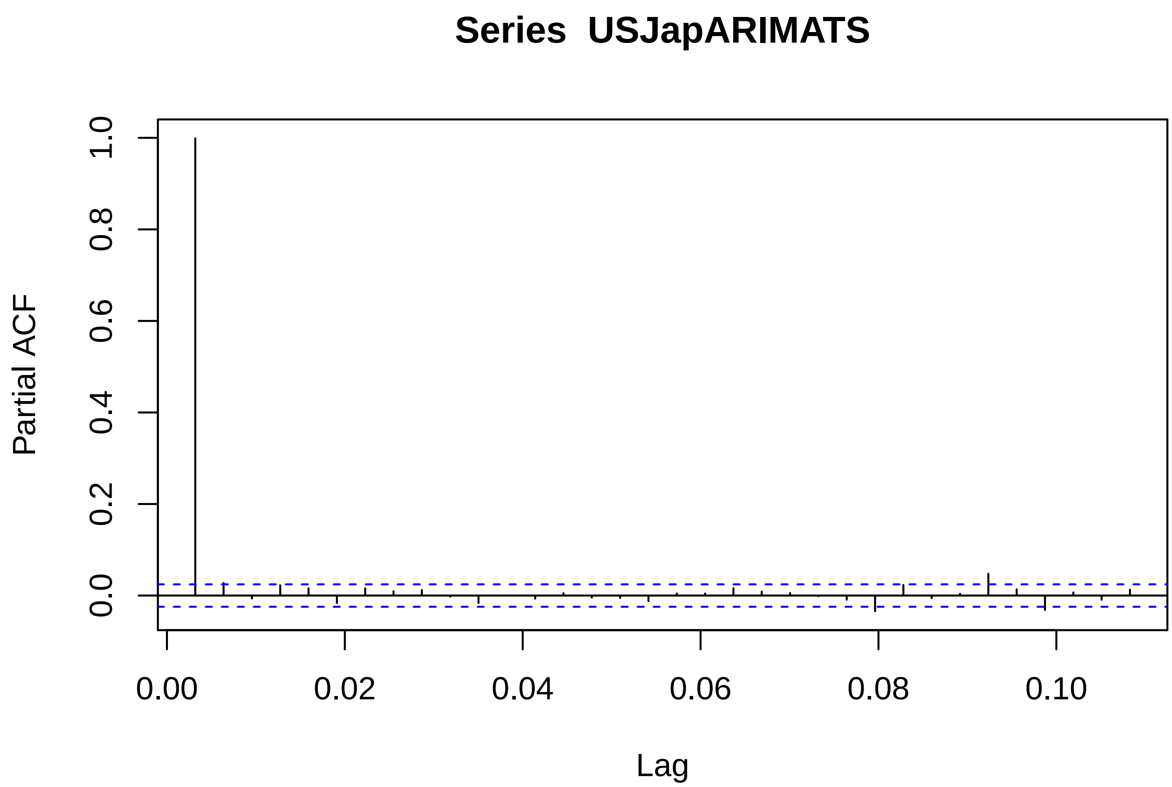


Calculating Autocorrlation function and partil autocorlation function

```
acf(USJapARIMATS,lag.max=34)
```

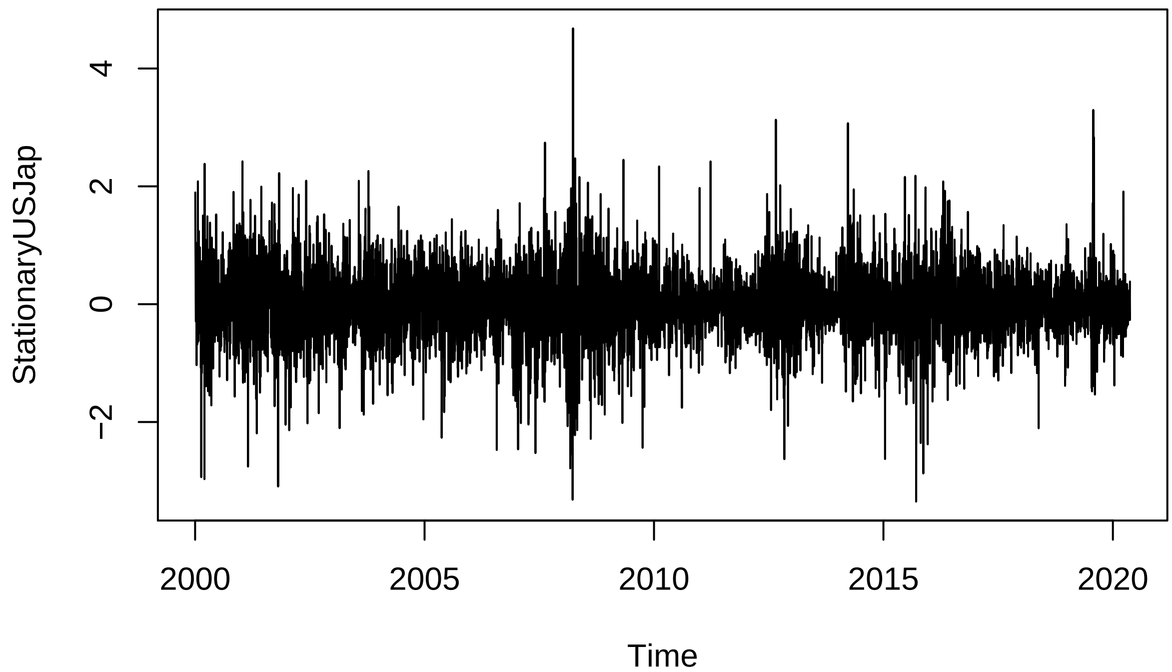



```r
pacf(USJapARIMATS, lag.max = 34)
```



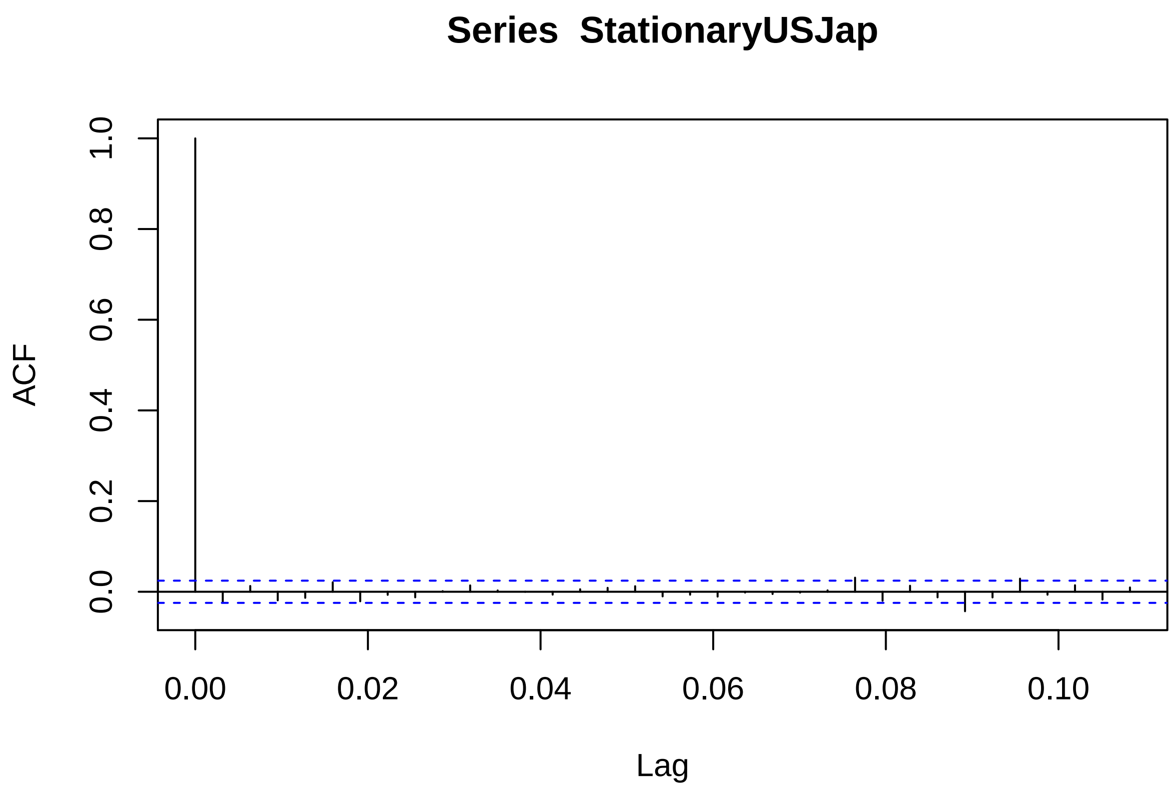
Adjusting and ensuring there are no seasonality

```
TSseasonallyadjustedUSJap <- USJapARIMATS- ComponentUSJapCurrency$seasonal
StationaryUSJap <- diff(TSseasonallyadjustedUSJap, differences=1)
plot(StationaryUSJap)
```

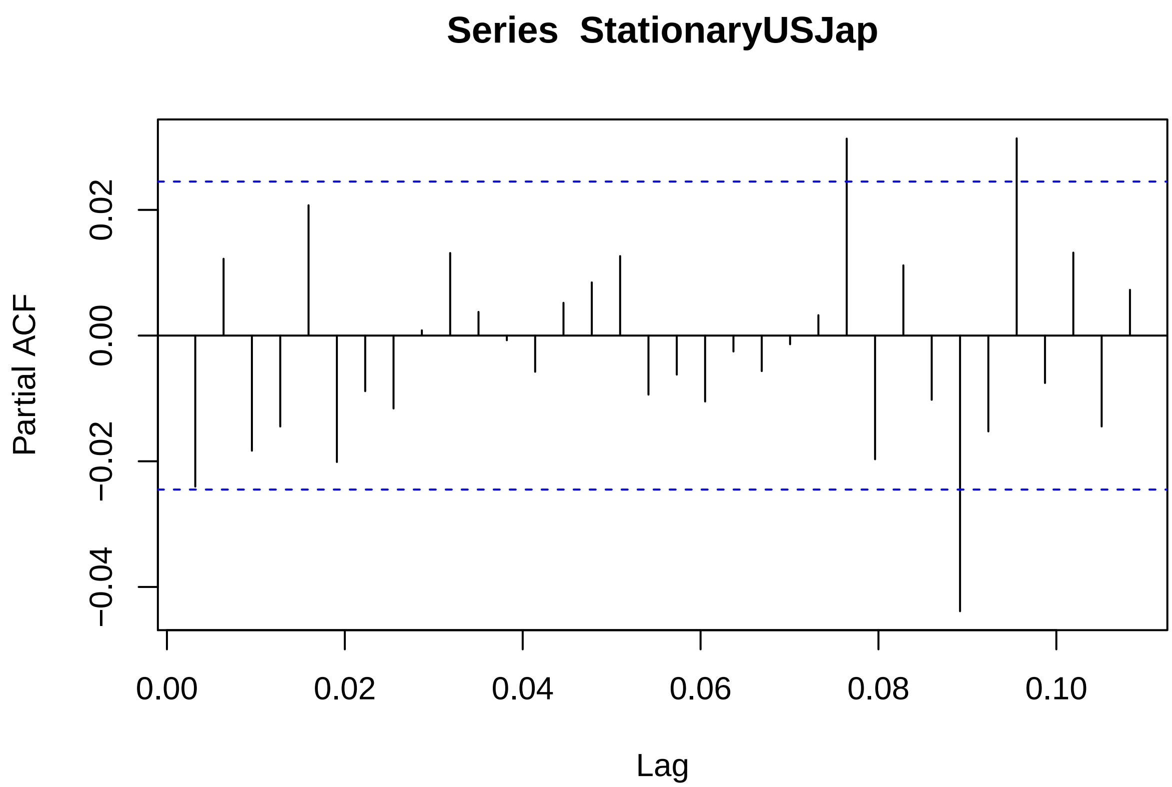


Calculating again for ACF and PACF after finding stationality

```
acf(StationaryUSJap, lag.max=34)
```




```
pacf(StationaryUSJap, lag.max=34)
```



Fitting The ARIMA Model

ARIMA fitting (1,1,0)

```
fitArima1USJap <- arima(USJapARIMATS, order =  c(0,1,0), include.mean = TRUE)
fitArima1USJap
```

```
##
## Call:
## arima(x = USJapARIMATS, order = c(0, 1, 0), include.mean = TRUE)
##
##
## sigma^2 estimated as 0.3649:  log likelihood = -5852.83,  aic = 11707.66
```

##Arima Fitting (1,1,1)

```
fitArima2USJap <- arima(USJapARIMATS, order =  c(1,1,1), include.mean = TRUE)
fitArima2USJap
```

```
##
## Call:
## arima(x = USJapARIMATS, order = c(1, 1, 1), include.mean = TRUE)
```

```
##
## Coefficients:
##          ar1      ma1
##       0.5962  -0.6160
## s.e.  0.6016   0.5936
##
## sigma^2 estimated as 0.3647:  log likelihood = -5850.91,  aic = 11707.82
```

Arima Fitting (2,1,1)

```r
fitArima3USJap <- arima(USJapARIMATS, order = c(2,1,1), include.mean = TRUE)
fitArima3USJap
```

```
##
## Call:
## arima(x = USJapARIMATS, order = c(2, 1, 1), include.mean = TRUE)
##
## Coefficients:
##          ar1     ar2      ma1
##       0.7354  0.0117  -0.7608
## s.e.  0.2878  0.0148   0.2878
##
## sigma^2 estimated as 0.3647:  log likelihood = -5850.42,  aic = 11708.85
```

##Fitting Arima (0,1,3)

```r
FitArima4USJap <- arima(USJapARIMATS, order = c(0,1,3), include.mean = TRUE)
FitArima4USJap
```

```
##
## Call:
## arima(x = USJapARIMATS, order = c(0, 1, 3), include.mean = TRUE)
##
## Coefficients:
##           ma1     ma2      ma3
##       -0.0253  0.0099  -0.0232
## s.e.   0.0125  0.0127   0.0127
##
## sigma^2 estimated as 0.3645:  log likelihood = -5848.97,  aic = 11705.93
```

##Best possible model is selected by AIC scores of the models

```r
library(dLagM)
```

```
## Warning: package 'dLagM' was built under R version 4.0.5
```

```
## Loading required package: nardl
```

```
## Warning: package 'nardl' was built under R version 4.0.5
```

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

```
## Loading required package: dynlm
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

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

```
ARIMAModelSelectionUSJAP = AIC(fitArima1USJap,fitArima2USJap,fitArima3USJap,FitArima4USJap)
sortScore(ARIMAModelSelectionUSJAP, score ="aic")
```

```
##                df      AIC
## FitArima4USJap  4 11705.93
## fitArima1USJap  1 11707.66
## fitArima2USJap  3 11707.82
## fitArima3USJap  4 11708.85
```

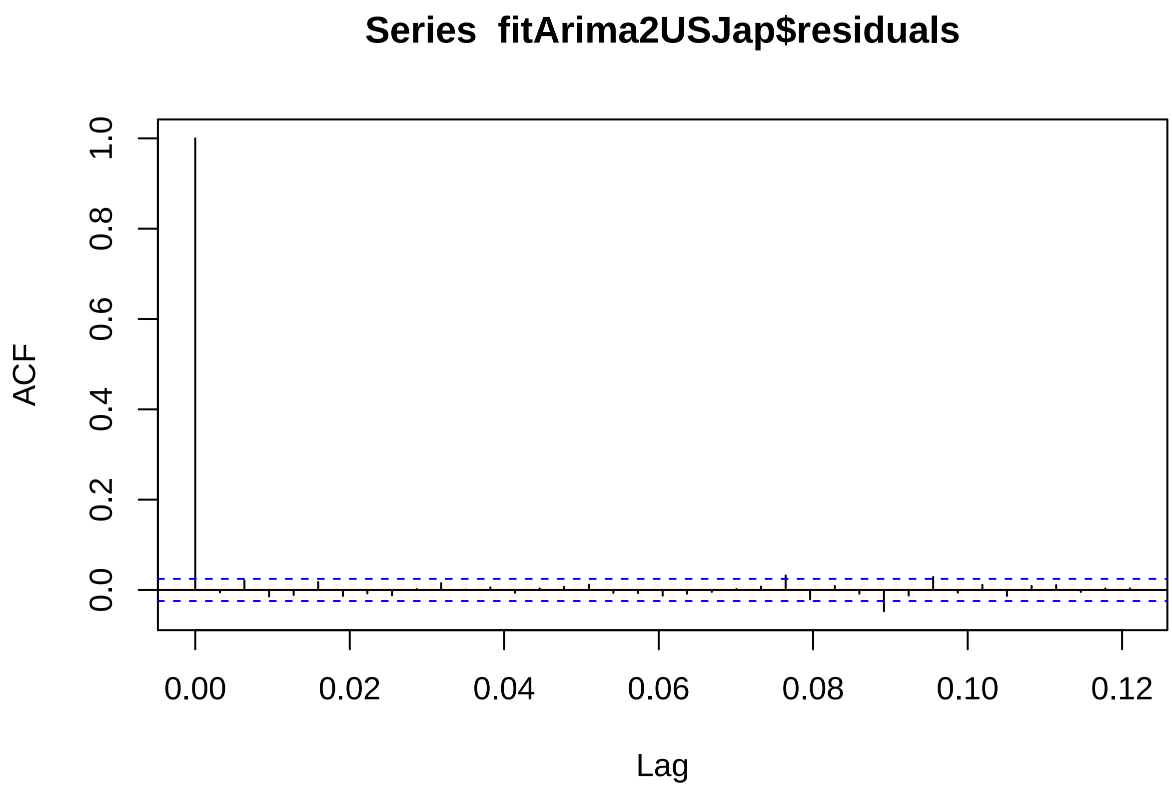
Base on the above the fitArima1CanJap is selected

```
confint(fitArima2USJap)
```

```
##          2.5 %    97.5 %
## ar1 -0.5828057 1.7752651
## ma1 -1.7793691 0.5473907
```

Runing code to obtain Box Test Rest

```
acf(fitArima2USJap$residuals)
```




```r
library(FitAR)
```

```
## Warning: package 'FitAR' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Loading required package: leaps
```

```
## Loading required package: ltsa
```

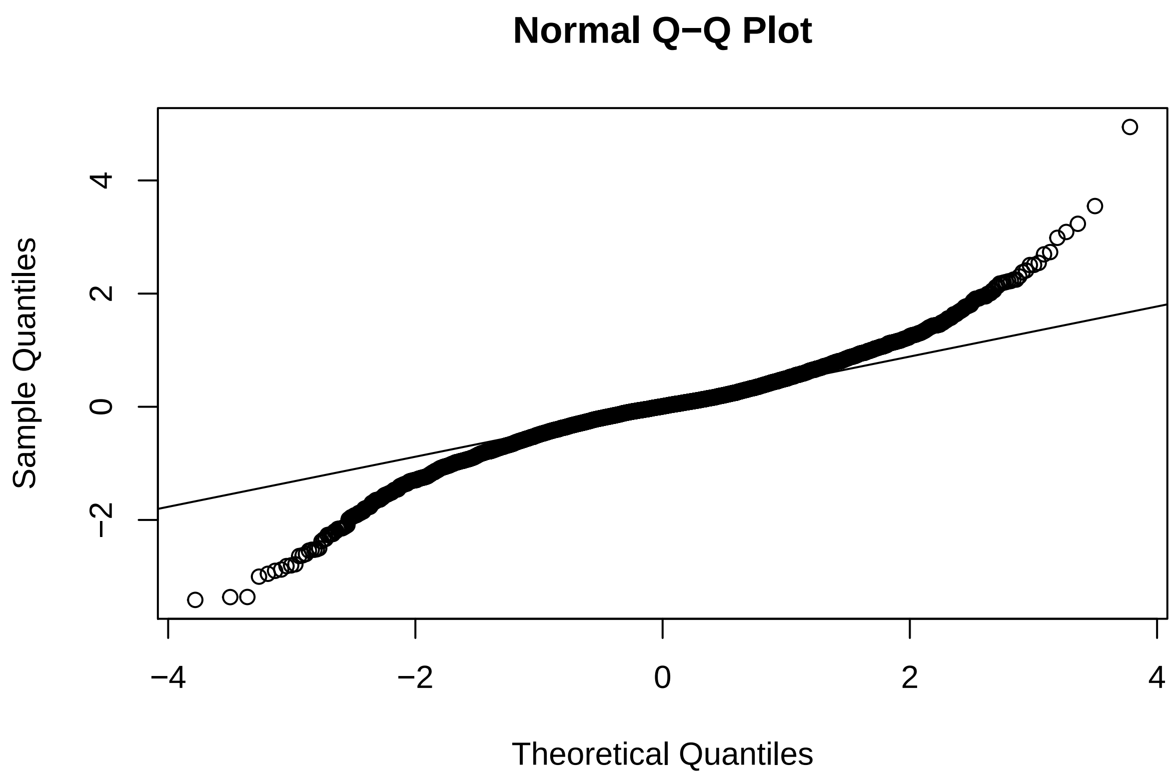
```
## Loading required package: bestglm
```

```
## Warning: package 'bestglm' was built under R version 4.0.5
```

```r
library(bestglm)
 Box.test(resid(fitArima2USJap),type="Ljung",lag=20,fitdf=1)
```

```
##
##  Box-Ljung test
##
## data:  resid(fitArima2USJap)
## X-squared = 14.779, df = 19, p-value = 0.7365
```

```r
qqnorm(fitArima2USJap$residuals)
qqline(fitArima2USJap$residuals)
```



Using Auto.arima to find the best model fit

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.0.5
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:FitAR':
##
##     BoxCox
```

```
## The following object is masked from 'package:dLagM':
##
##     forecast
```

```r
auto.arima(USJapARIMATS, trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[314] with drift         : Inf
##  ARIMA(0,1,0)             with drift         : 11710.67
##  ARIMA(1,1,0)(1,0,0)[314] with drift         : Inf
##  ARIMA(0,1,1)(0,0,1)[314] with drift         : Inf
##  ARIMA(0,1,0)                                : 11708.67
##  ARIMA(0,1,0)(1,0,0)[314] with drift         : 11593.23
##  ARIMA(0,1,0)(2,0,0)[314] with drift         : Inf
##  ARIMA(0,1,0)(1,0,1)[314] with drift         : Inf
##  ARIMA(0,1,0)(0,0,1)[314] with drift         : 11711.89
##  ARIMA(0,1,0)(2,0,1)[314] with drift         : Inf
##  ARIMA(0,1,1)(1,0,0)[314] with drift         : Inf
##  ARIMA(1,1,1)(1,0,0)[314] with drift         : Inf
##  ARIMA(0,1,0)(1,0,0)[314]                    : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,1,0)(1,0,0)[314] with drift         : Inf
##  ARIMA(0,1,0)                                : 11707.66
##
##  Best model: ARIMA(0,1,0)
```

```
## Series: USJapARIMATS
## ARIMA(0,1,0)
##
## sigma^2 estimated as 0.3649:  log likelihood=-5852.83
## AIC=11707.66   AICc=11707.66   BIC=11714.42
```

forecasting using Best model: ARIMA(1,1,0)

```r
forecastarimaUSJa<- predict(fitArima2USJap,n.ahead = 100)
forecastarimaUSJa
```

```
## $pred
## Time Series:
## Start = c(2020, 119)
## End = c(2020, 218)
## Frequency = 314
##   [1] 103.2618 103.2634 103.2644 103.2650 103.2653 103.2655 103.2656 103.2657
##   [9] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [17] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [25] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [33] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [41] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [49] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [57] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
```

```
##  [65] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [73] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [81] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [89] 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658 103.2658
##  [97] 103.2658 103.2658 103.2658 103.2658
##
## $se
## Time Series:
## Start = c(2020, 119)
## End = c(2020, 218)
## Frequency = 314
##   [1] 0.6039249 0.8456827 1.0282315 1.1808456 1.3147631 1.4356414 1.5467691
##   [8] 1.6502442 1.7474981 1.8395581 1.9271918 2.0109909 2.0914239 2.1688698
##  [15] 2.2436403 2.3159960 2.3861575 2.4543133 2.5206265 2.5852391 2.6482756
##  [22] 2.7098460 2.7700482 2.8289695 2.8866884 2.9432756 2.9987952 3.0533054
##  [29] 3.1068594 3.1595057 3.2112891 3.2622506 3.3124281 3.3618568 3.4105693
##  [36] 3.4585957 3.5059642 3.5527013 3.5988314 3.6443777 3.6893618 3.7338039
##  [43] 3.7777232 3.8211378 3.8640646 3.9065197 3.9485184 3.9900751 4.0312033
##  [50] 4.0719162 4.1122260 4.1521445 4.1916829 4.2308517 4.2696613 4.3081213
##  [57] 4.3462409 4.3840291 4.4214944 4.4586449 4.4954883 4.5320323 4.5682839
##  [64] 4.6042501 4.6399375 4.6753525 4.7105013 4.7453898 4.7800236 4.8144082
##  [71] 4.8485490 4.8824511 4.9161194 4.9495587 4.9827736 5.0157685 5.0485478
##  [78] 5.0811157 5.1134761 5.1456330 5.1775902 5.2093513 5.2409200 5.2722997
##  [85] 5.3034937 5.3345053 5.3653376 5.3959938 5.4264768 5.4567895 5.4869347
##  [92] 5.5169152 5.5467337 5.5763928 5.6058949 5.6352426 5.6644382 5.6934841
##  [99] 5.7223826 5.7511359
```

```r
par(mfrow = c(1,1))
```