# ARIMA Model GBP And JPY

Jane

28/04/2021

## Forcasting Exchange Rate Using ARIMA Model for Bristish Pound And Japanese Yen

### Reading GBP and JPY 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
```

```
GBPJPYARIMA <-  read.csv ("GBPJPY_Candlestick_1_D_BID_01.01.2000-31.12.2020.csv")%>%
  select('GMT.TIME', CLOSE)%>%
  rename(Date = ('GMT.TIME'), RateGBPJPY = ("CLOSE"))


head(GBPJPYARIMA)
```

```
##         Date RateGBPJPY
## 1 2000-01-03     166.01
## 2 2000-01-04     168.81
## 3 2000-01-05     171.34
## 4 2000-01-06     173.37
## 5 2000-01-07     172.56
## 6 2000-01-10     171.98
```

### 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
GBPJPYARIMA$Date <- lubridate::ymd(GBPJPYARIMA$Date)
head(GBPJPYARIMA)
```

```
##         Date RateGBPJPY
## 1 2000-01-03     166.01
## 2 2000-01-04     168.81
## 3 2000-01-05     171.34
## 4 2000-01-06     173.37
## 5 2000-01-07     172.56
## 6 2000-01-10     171.98
```

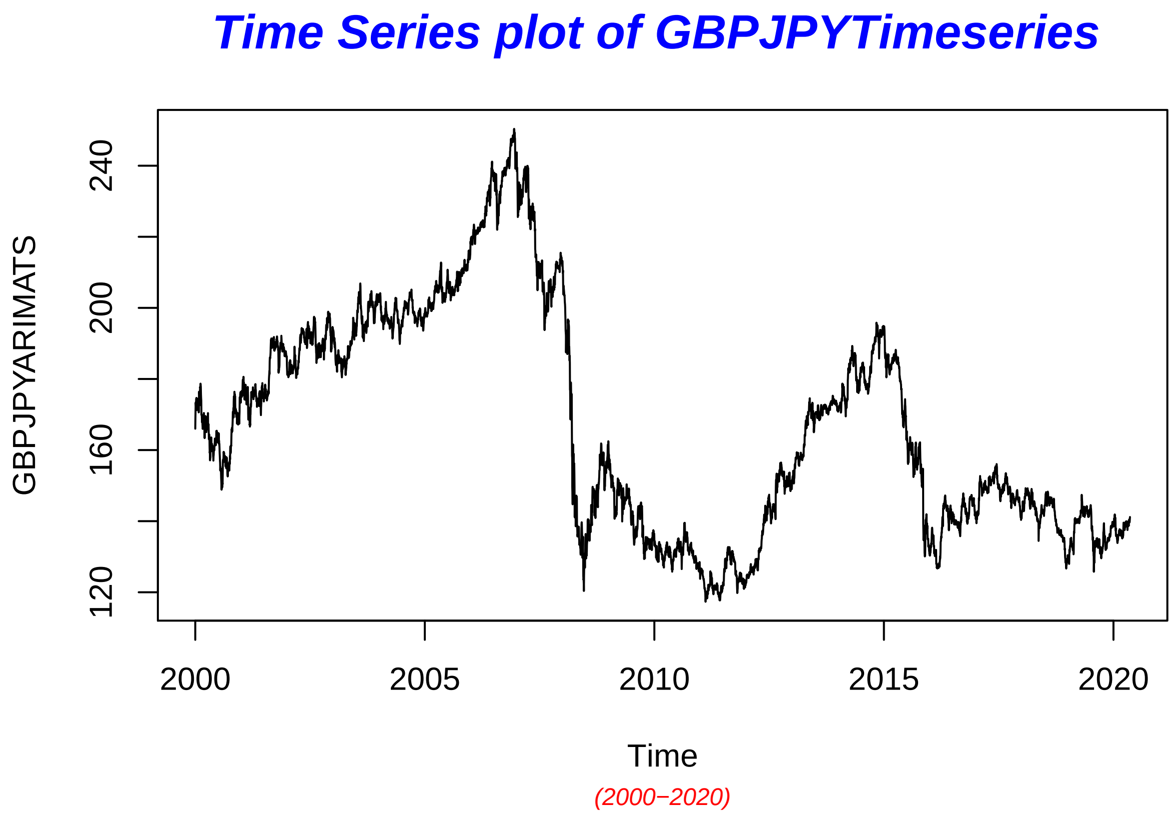
##Checking for obvious errors or missingg value

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

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

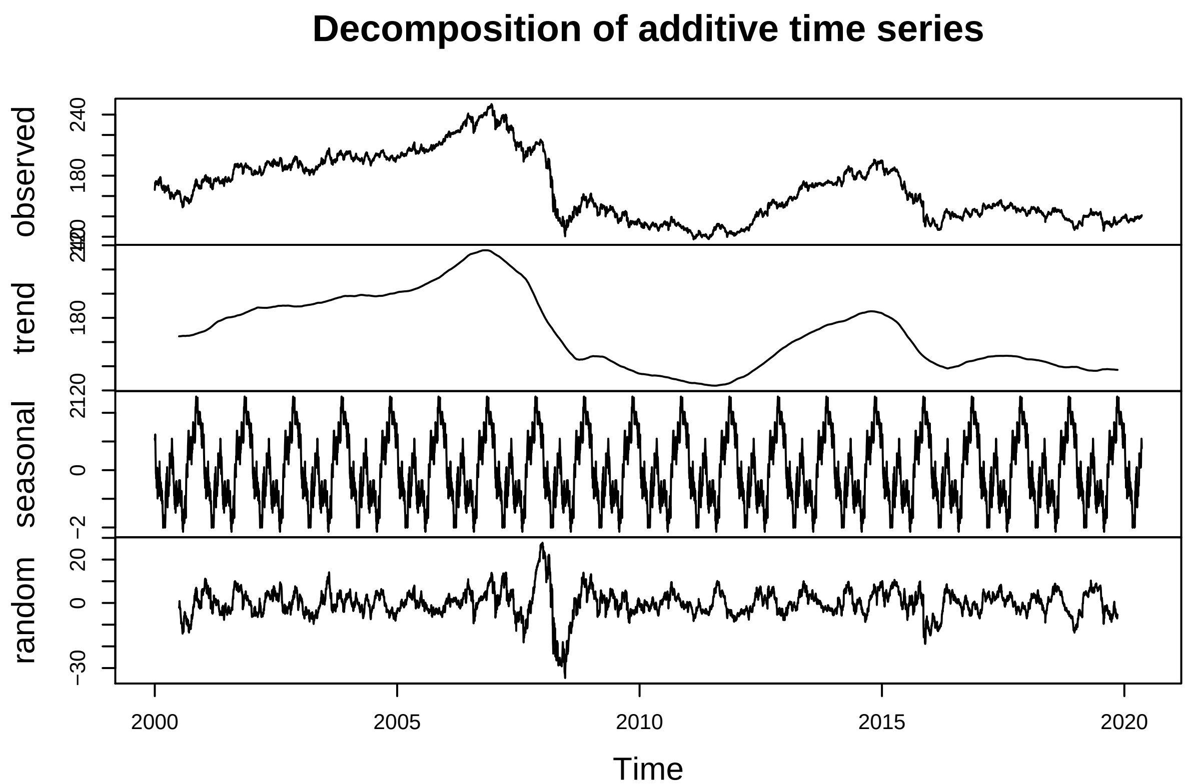
##Converting the data set into time series object

```r
#Converting the data set into time series object
GBPJPYARIMATS<- ts(as.vector(GBPJPYARIMA$Rate),  frequency = 314, start= c(2000,01,03))
plot.ts(GBPJPYARIMATS)
title("Time Series plot of GBPJPYTimeseries ", 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

```
ComponentGBPJPY <- decompose(GBPJPYARIMATS)
plot(ComponentGBPJPY)
```




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

```r
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
```

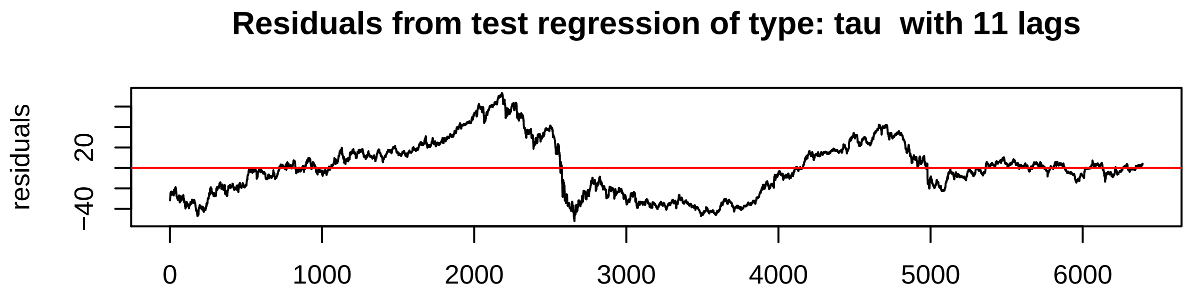
```
## Loading required package: timeSeries
```

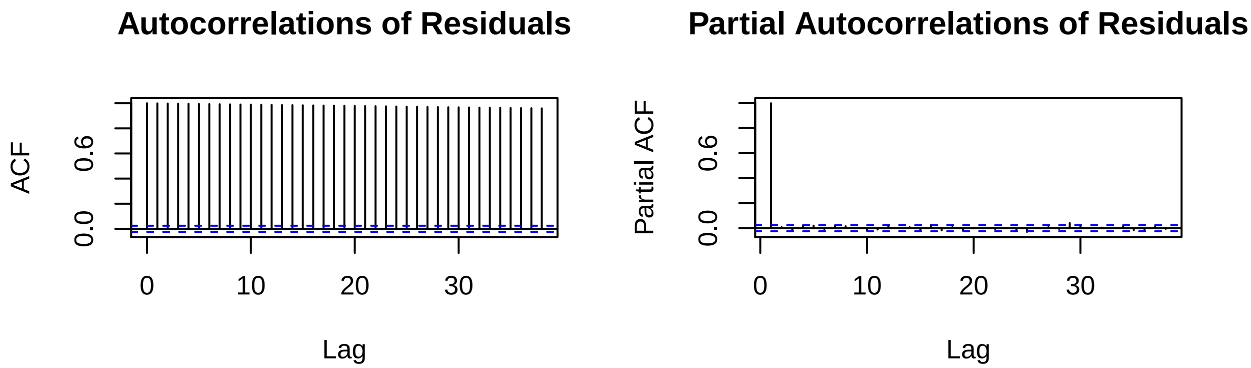
```
## Warning: package 'timeSeries' was built under R version 4.0.5
```

```
## Loading required package: fBasics
```

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

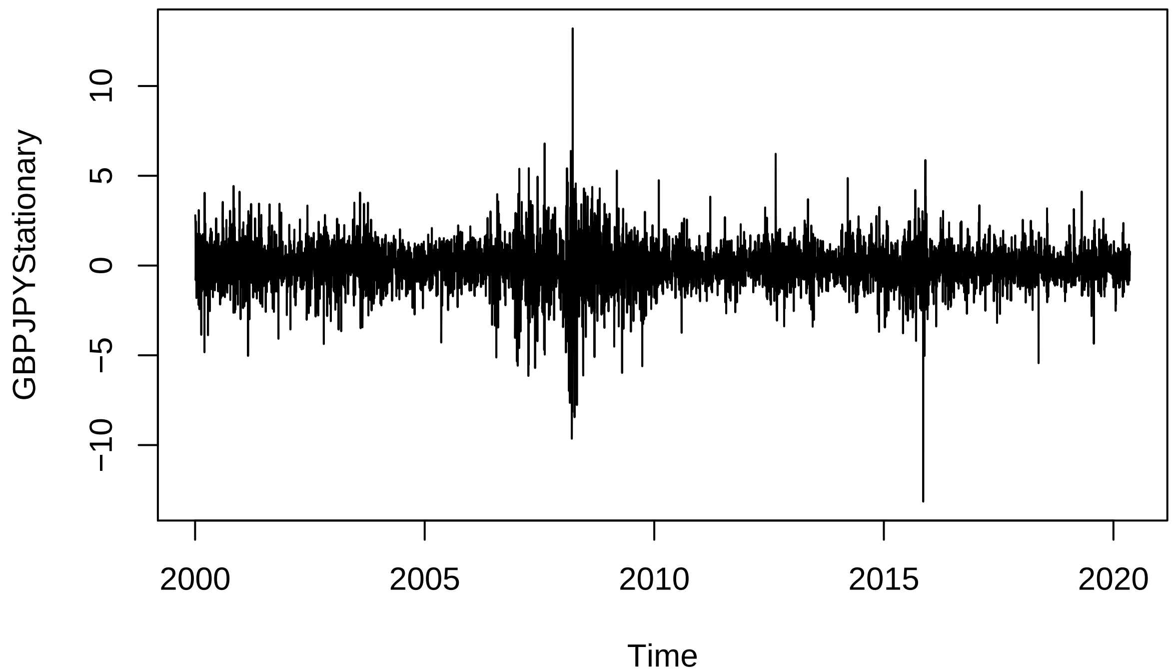
```r
urkpssTest(GBPJPYARIMATS, type = c("tau"), lags = c("short"),use.lag = NULL, doplot = TRUE)
```



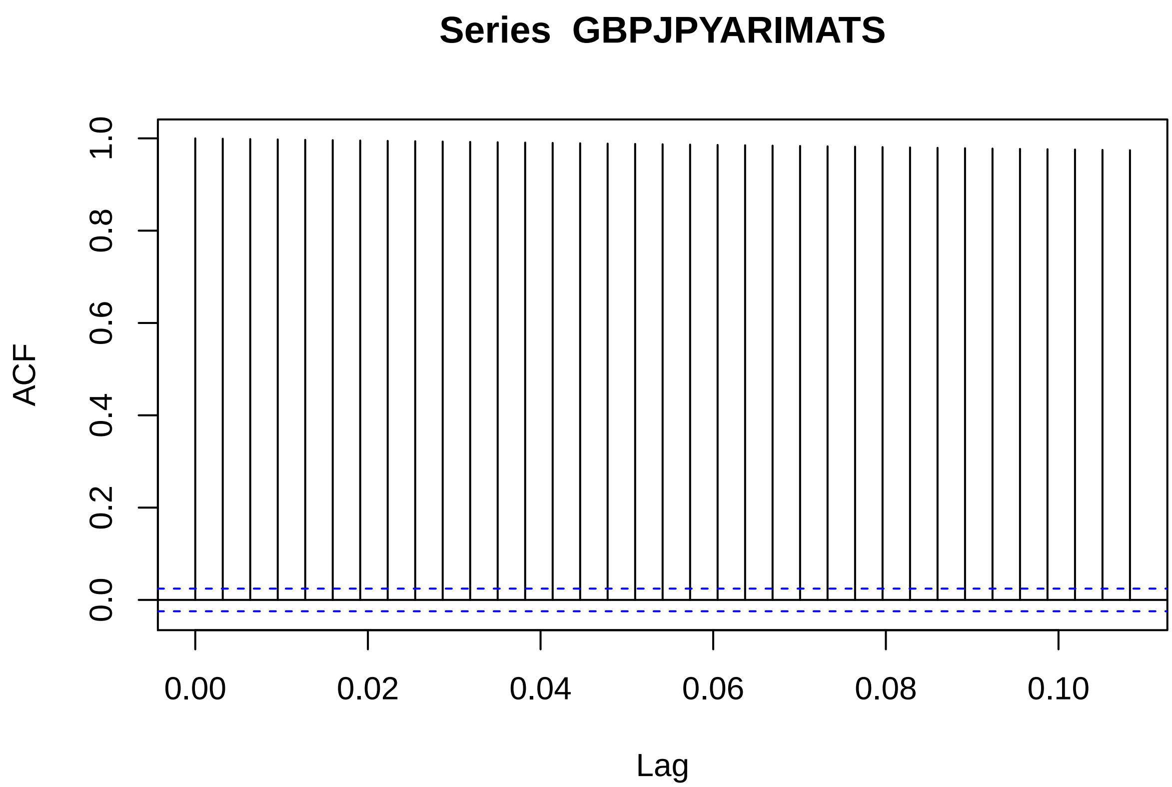

```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Tue May 04 00:13:38 2021 by user: janeo
```

```r
GBPJPYStationary= diff(GBPJPYARIMATS, differences=1)
plot(GBPJPYStationary)
```

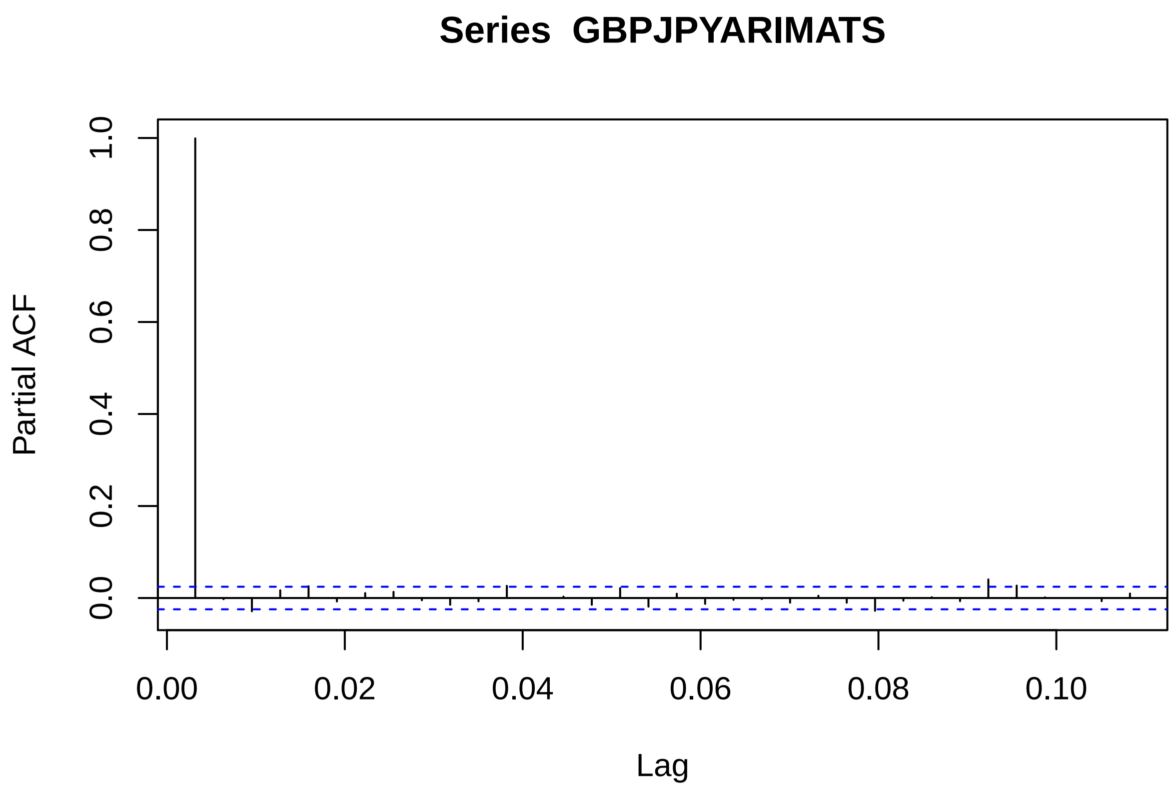



### Calculating Autocorrlation function and partil autocorlation function

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

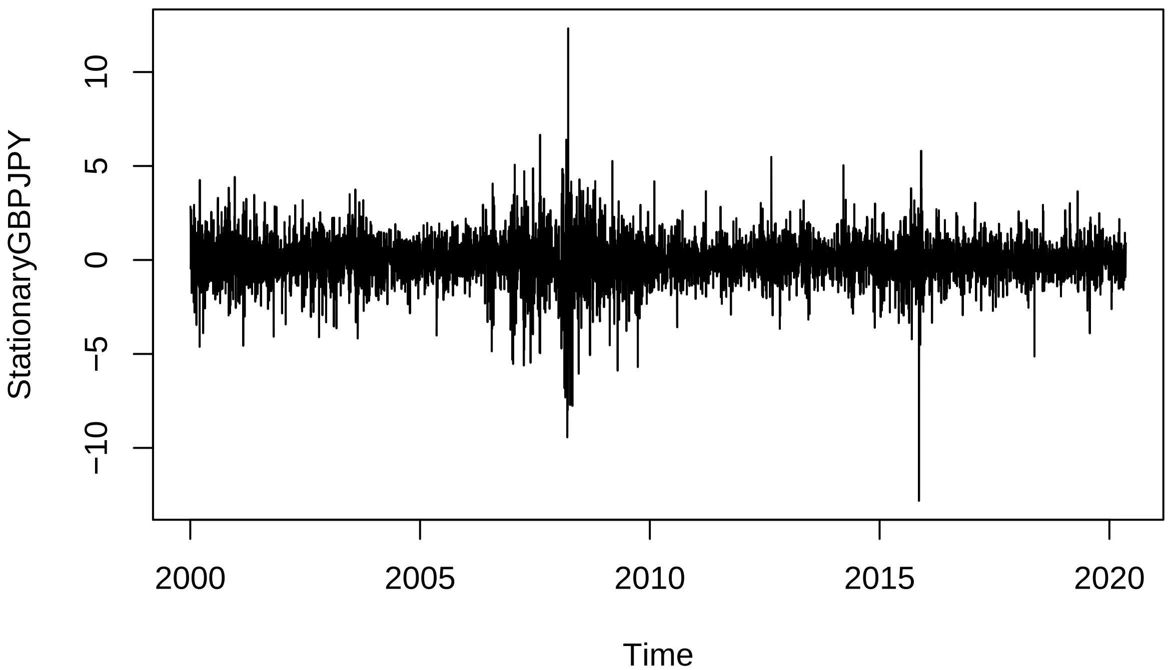



```r
pacf(GBPJPYARIMATS, lag.max = 34)
```



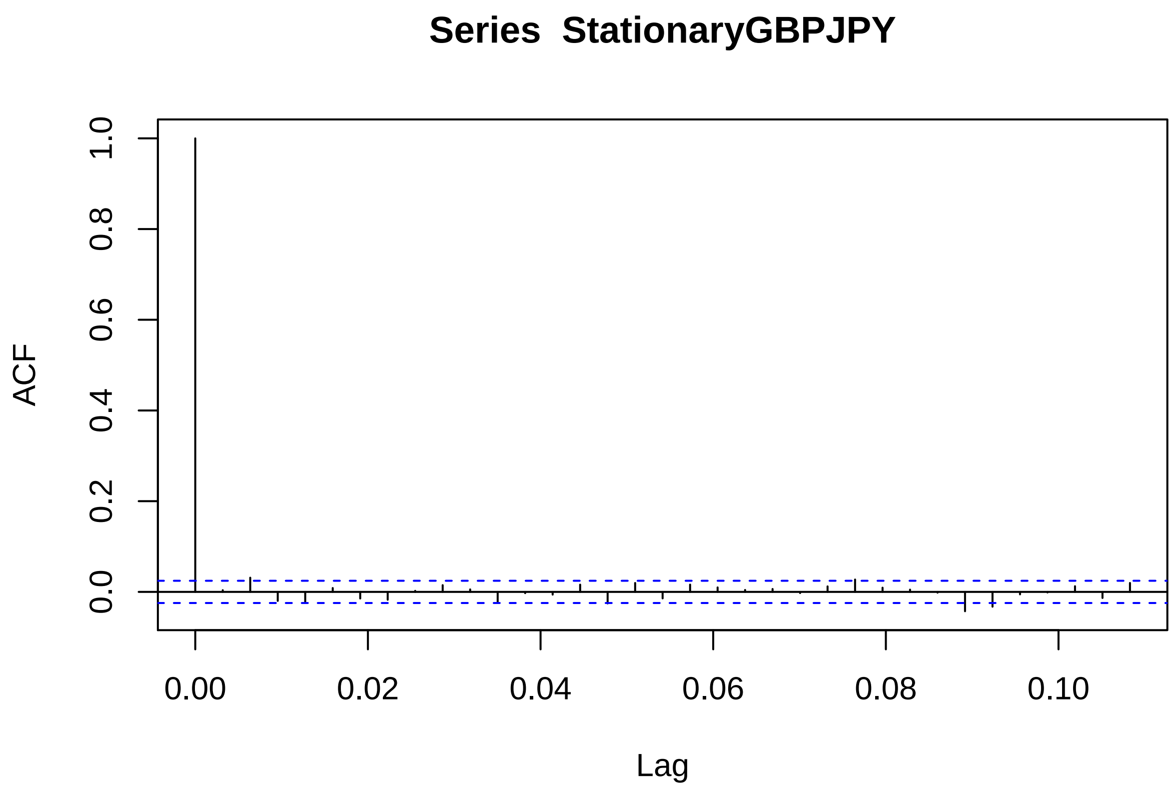

### Adjusting and ensuring there are no seasonality

```
TSseasonallyadjustedGBPJPY <- GBPJPYARIMATS- ComponentGBPJPY$seasonal
StationaryGBPJPY<- diff(TSseasonallyadjustedGBPJPY, differences=1)
plot(StationaryGBPJPY)
```

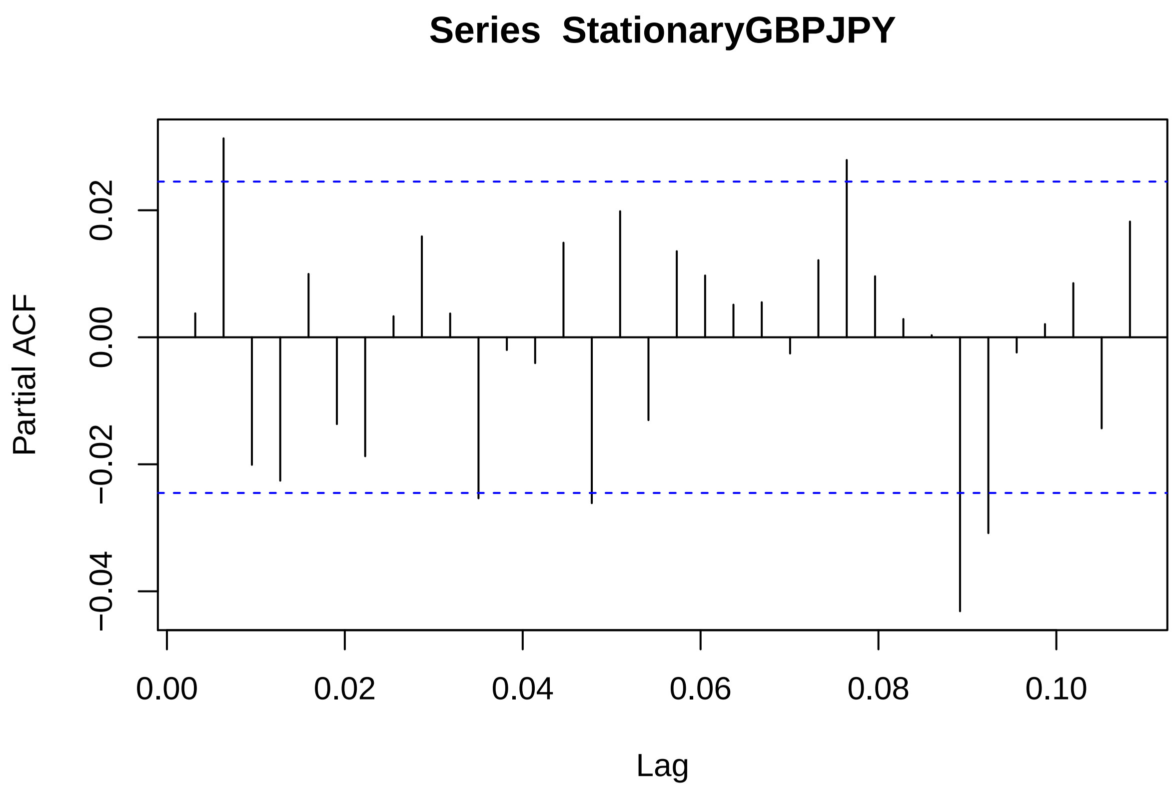



### Calculating again for ACF and PACF after finding stationality

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




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




## Fitting The ARIMA Model

### ARIMA fitting (1,1,0)

```
fitArima1GBPJPY <- arima(GBPJPYARIMATS, order =  c(1,1,0), include.mean = TRUE)
fitArima1GBPJPY
```

```
##
## Call:
## arima(x = GBPJPYARIMATS, order = c(1, 1, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1
##       0.0009
## s.e.  0.0125
##
## sigma^2 estimated as 1.416:  log likelihood = -10183.86,  aic = 20371.72
```

##Arima Fitting (0,1,0)

```
fitArima2GBPJPY <- arima(GBPJPYARIMATS, order =  c(0,1,0), include.mean = TRUE)
fitArima2GBPJPY
```

```
##
## Call:
## arima(x = GBPJPYARIMATS, order = c(0, 1, 0), include.mean = TRUE)
##
##
## sigma^2 estimated as 1.416:  log likelihood = -10183.86,  aic = 20369.73
```

### Arima Fitting (2,1,1)

```r
fitArima3GBPJPY <- arima(GBPJPYARIMATS, order = c(2,1,1), include.mean = TRUE)
fitArima3GBPJPY
```

```
##
## Call:
## arima(x = GBPJPYARIMATS, order = c(2, 1, 1), include.mean = TRUE)
##
## Coefficients:
##           ar1     ar2     ma1
##       -0.2093  0.0301  0.2104
## s.e.   0.2288  0.0126  0.2286
##
## sigma^2 estimated as 1.415:  log likelihood = -10180.87,  aic = 20369.74
```

##Fitting Arima (0,1,3)

```r
fitArima4GBPJPY <- arima(GBPJPYARIMATS, order = c(3,1,0), include.mean = TRUE)
fitArima4GBPJPY
```

```
##
## Call:
## arima(x = GBPJPYARIMATS, order = c(3, 1, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1     ar2      ar3
##       0.0014  0.0286  -0.0185
## s.e.  0.0125  0.0125   0.0125
##
## sigma^2 estimated as 1.415:  log likelihood = -10180.15,  aic = 20368.3
```

##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
```

```
ARIMAModelSelectionGBPJPY = AIC(fitArima1GBPJPY,fitArima2GBPJPY,fitArima3GBPJPY,fitArima4GBPJPY)
sortScore(ARIMAModelSelectionGBPJPY, score ="aic")
```

```
##                 df      AIC
## fitArima4GBPJPY  4 20368.30
## fitArima2GBPJPY  1 20369.73
## fitArima3GBPJPY  4 20369.74
## fitArima1GBPJPY  2 20371.72
```

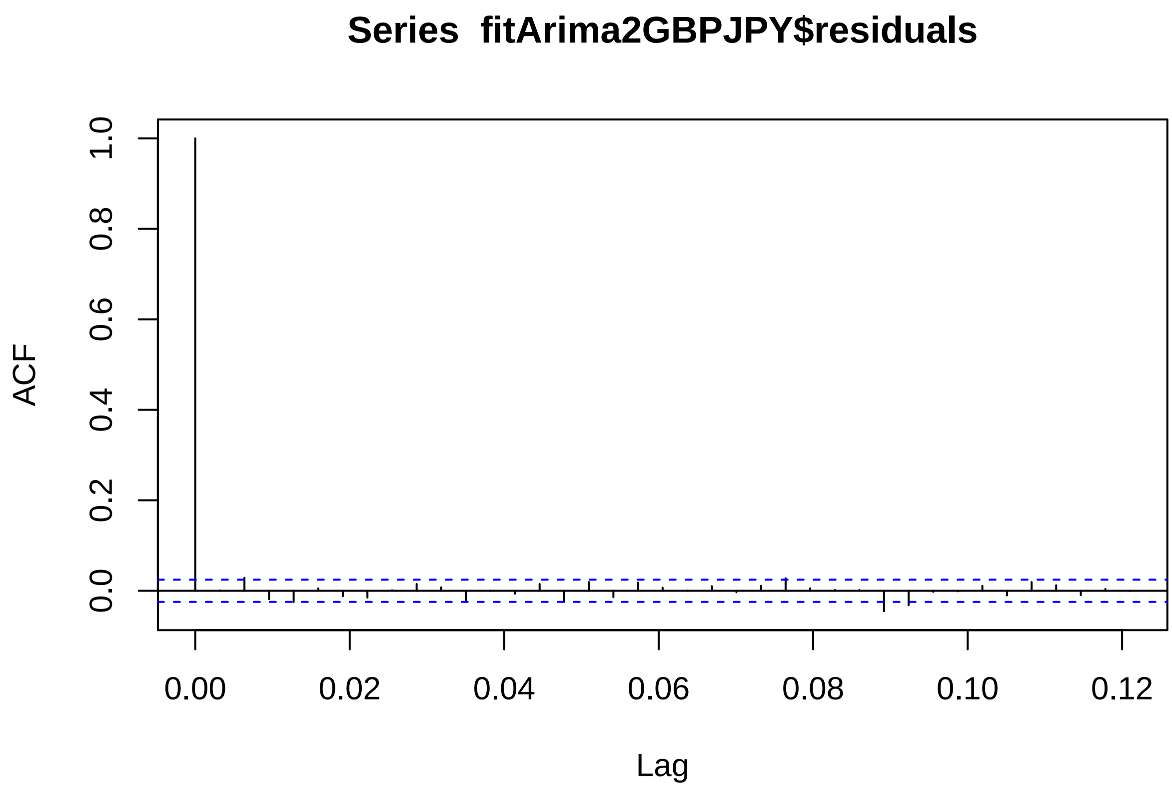
**Base on the above the fitArima1CanJap is selected**

```
confint(fitArima2GBPJPY)
```

```
##      2.5 % 97.5 %
```

## Runing code to obtain Box Test Rest

```
acf(fitArima2GBPJPY$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
```

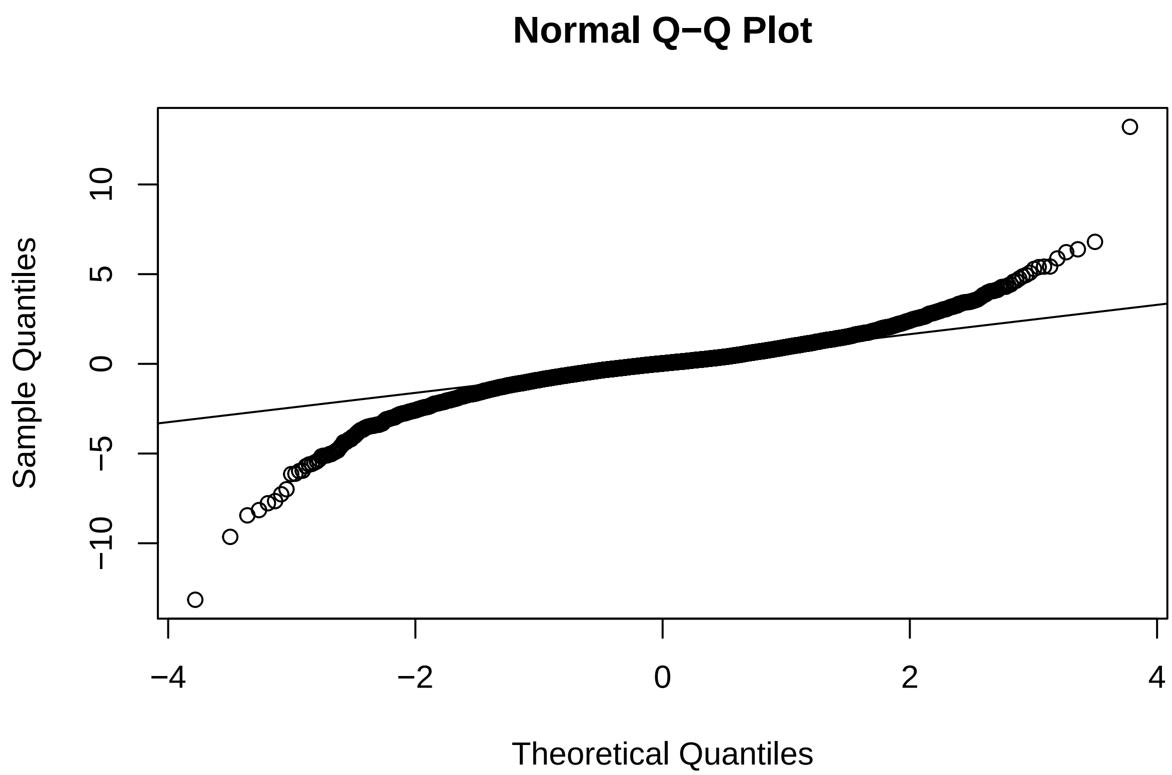
```
## Loading required package: bestglm
```

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

```r
library(bestglm)
 Box.test(resid(fitArima2GBPJPY),type="Ljung",lag=20,fitdf=1)
```

```
##
##  Box-Ljung test
##
## data:  resid(fitArima2GBPJPY)
## X-squared = 31.293, df = 19, p-value = 0.03748
```

```r
qqnorm(fitArima2GBPJPY$residuals)
qqline(fitArima2GBPJPY$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(GBPJPYARIMATS, 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         : 20371.32
##  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)                                : 20369.38
##  ARIMA(0,1,0)(1,0,0)[314] with drift         : 20286.36
##  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         : Inf
##  ARIMA(0,1,0)(2,0,1)[314] with drift         : Inf
##  ARIMA(0,1,1)(1,0,0)[314] with drift         : 20288.36
##  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         : 20372.69
##
##  Best model: ARIMA(0,1,0)(1,0,0)[314] with drift
```

```
## Series: GBPJPYARIMATS
## ARIMA(0,1,0)(1,0,0)[314] with drift
##
## Coefficients:
##         sar1    drift
##       0.0125  -0.0041
## s.e.  0.0127   0.0151
##
## sigma^2 estimated as 1.417:  log likelihood=-10183.34
## AIC=20372.69   AICc=20372.69   BIC=20392.98
```

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

```r
forecastarimaGBPJPY<- predict(fitArima2GBPJPY,n.ahead = 100)
forecastarimaGBPJPY
```

```
## $pred
## Time Series:
## Start = c(2020, 115)
## End = c(2020, 214)
## Frequency = 314
##   [1] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [10] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [19] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [28] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [37] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
```

```
##  [46] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [55] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [64] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [73] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [82] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
##  [91] 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168 141.168
## [100] 141.168
##
## $se
## Time Series:
## Start = c(2020, 115)
## End = c(2020, 214)
## Frequency = 314
##   [1]  1.190094  1.683048  2.061304  2.380189  2.661132  2.915124  3.148694
##   [8]  3.366095  3.570283  3.763409  3.947097  4.122608  4.290947  4.452926
##  [15]  4.609216  4.760378  4.906885  5.049143  5.187501  5.322264  5.453698
##  [22]  5.582038  5.707492  5.830248  5.950472  6.068315  6.183912  6.297388
##  [29]  6.408855  6.518416  6.626165  6.732191  6.836572  6.939383  7.040694
##  [36]  7.140567  7.239062  7.336235  7.432137  7.526818  7.620323  7.712693
##  [43]  7.803971  7.894193  7.983396  8.071613  8.158876  8.245216  8.330661
##  [50]  8.415238  8.498974  8.581893  8.664018  8.745372  8.825977  8.905851
##  [57]  8.985016  9.063489  9.141289  9.218432  9.294935  9.370813  9.446082
##  [64]  9.520756  9.594848  9.668373  9.741343  9.813770  9.885667  9.957044
##  [71] 10.027914 10.098286 10.168171 10.237579 10.306520 10.375003 10.443036
##  [78] 10.510629 10.577791 10.644528 10.710850 10.776764 10.842276 10.907396
##  [85] 10.972129 11.036482 11.100462 11.164075 11.227328 11.290227 11.352777
##  [92] 11.414985 11.476855 11.538394 11.599606 11.660496 11.721071 11.781334
##  [99] 11.841290 11.900944
```

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