ARIMA Model GBP And USD

Jane

28/04/2021

Forcasting Exchange Rate Using ARIMA Model for Bristish Pound And US Dollar

Reading GBP and USD 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
```

```
GBPUSDARIMA <-  read.csv ("GBPUSD_Candlestick_1_D_BID_01.01.2000-31.12.2020.csv")%>%
  select('GMT.TIME', CLOSE)%>%
  rename(Date = ('GMT.TIME'), RateGBPUSD = ("CLOSE"))


head(GBPUSDARIMA)
```

```
##         Date RateGBPUSD
## 1 2000-01-03     1.6355
## 2 2000-01-04     1.6357
## 3 2000-01-05     1.6423
## 4 2000-01-06     1.6469
## 5 2000-01-07     1.6391
## 6 2000-01-10     1.6369
```

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
GBPUSDARIMA$Date <- lubridate::ymd(GBPUSDARIMA$Date)
head(GBPUSDARIMA)
```

```
##         Date RateGBPUSD
## 1 2000-01-03     1.6355
## 2 2000-01-04     1.6357
## 3 2000-01-05     1.6423
## 4 2000-01-06     1.6469
## 5 2000-01-07     1.6391
## 6 2000-01-10     1.6369
```

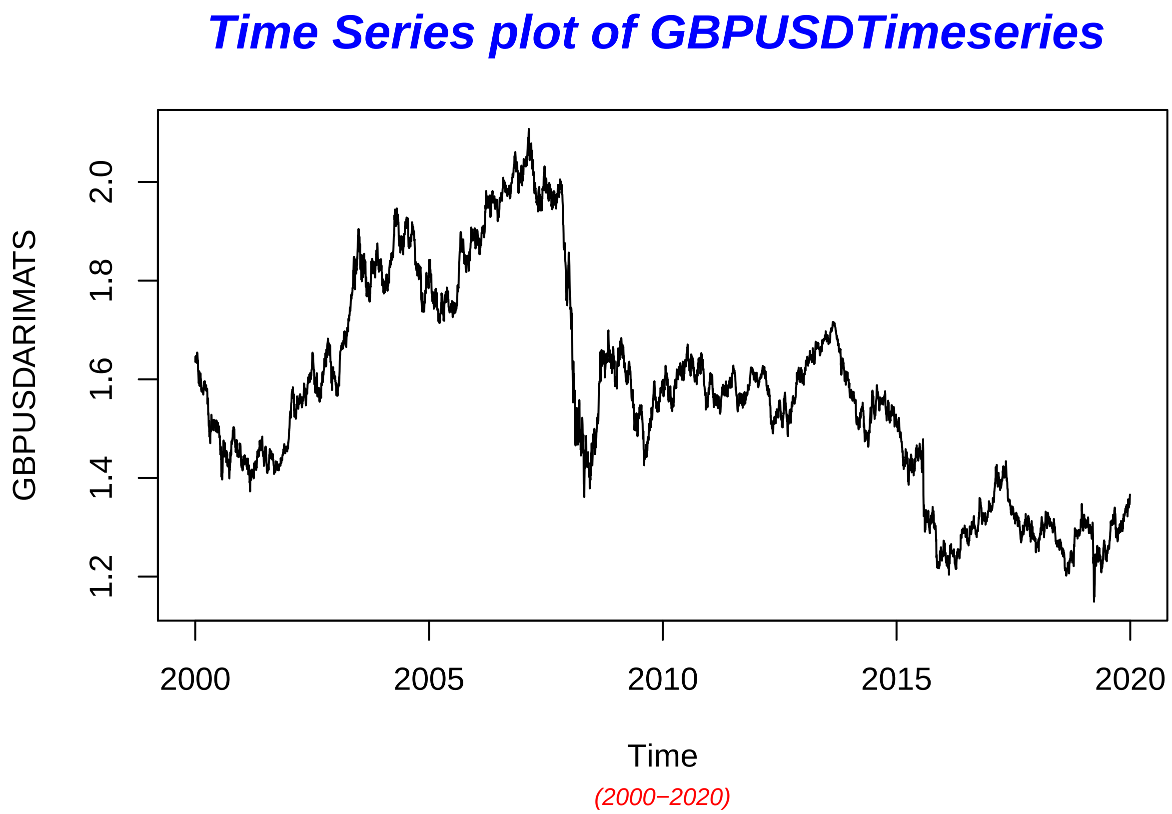
##Checking for obvious errors or missingg value

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

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

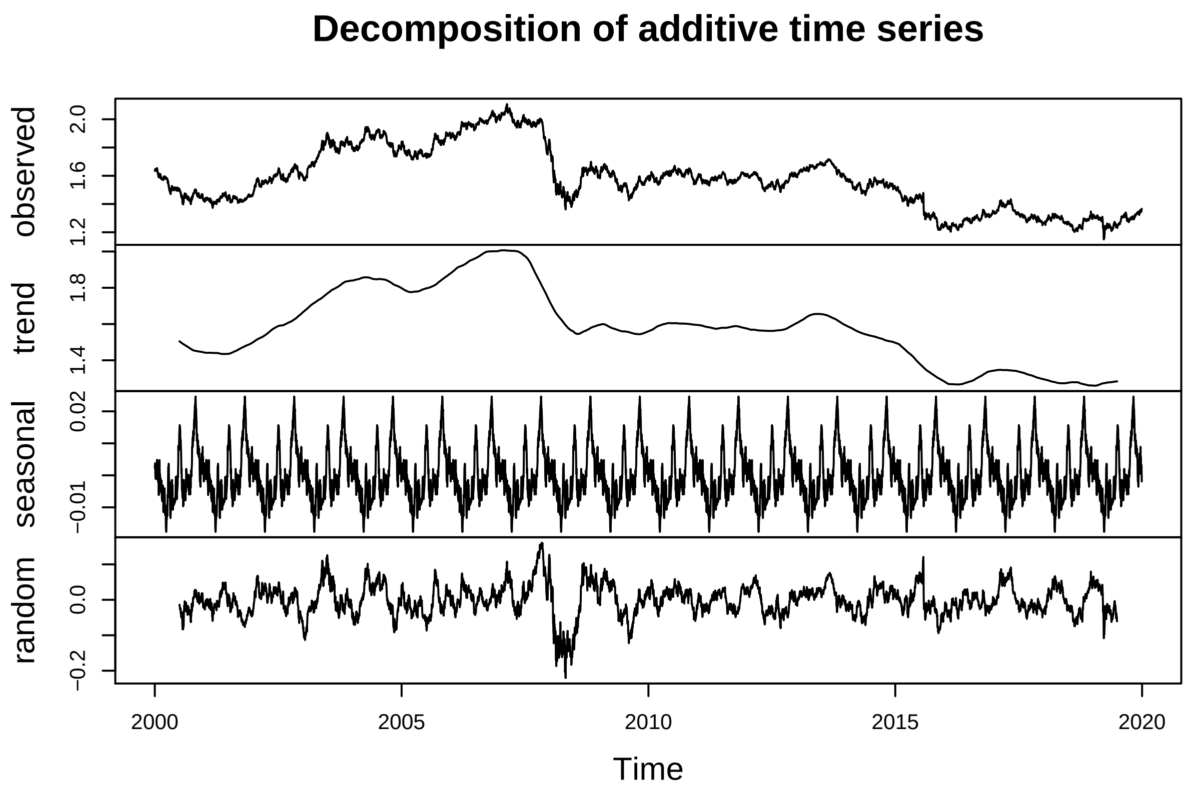
##Converting the data set into time series object

```r
#Converting the data set into time series object
GBPUSDARIMATS<- ts(as.vector(GBPUSDARIMA$Rate),  frequency = 320
                   , start= c(2000,01,03))
plot.ts(GBPUSDARIMATS)
title("Time Series plot of GBPUSDTimeseries ", 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

```
ComponentGBPUSD <- decompose(GBPUSDARIMATS)
plot(ComponentGBPUSD)
```

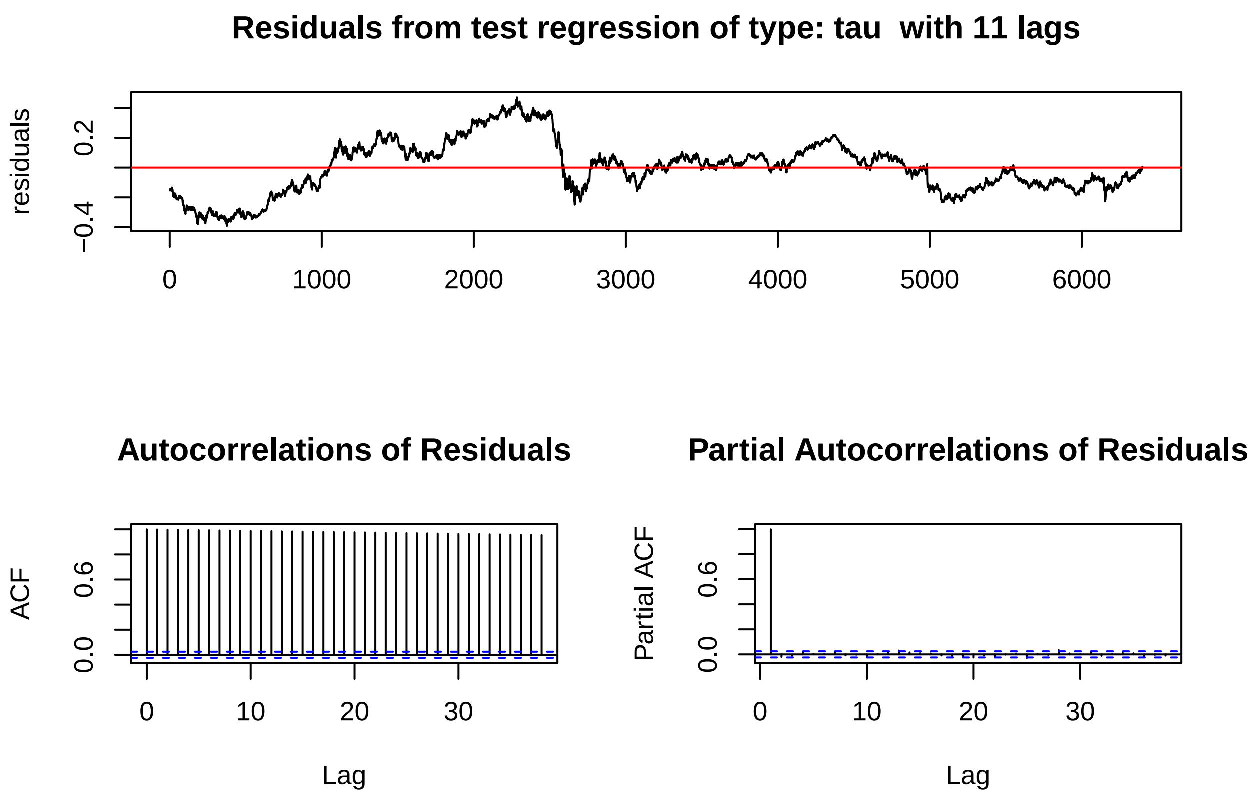


To To achieve stationarity by differencing the data – compute the differences between consecutive observations

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

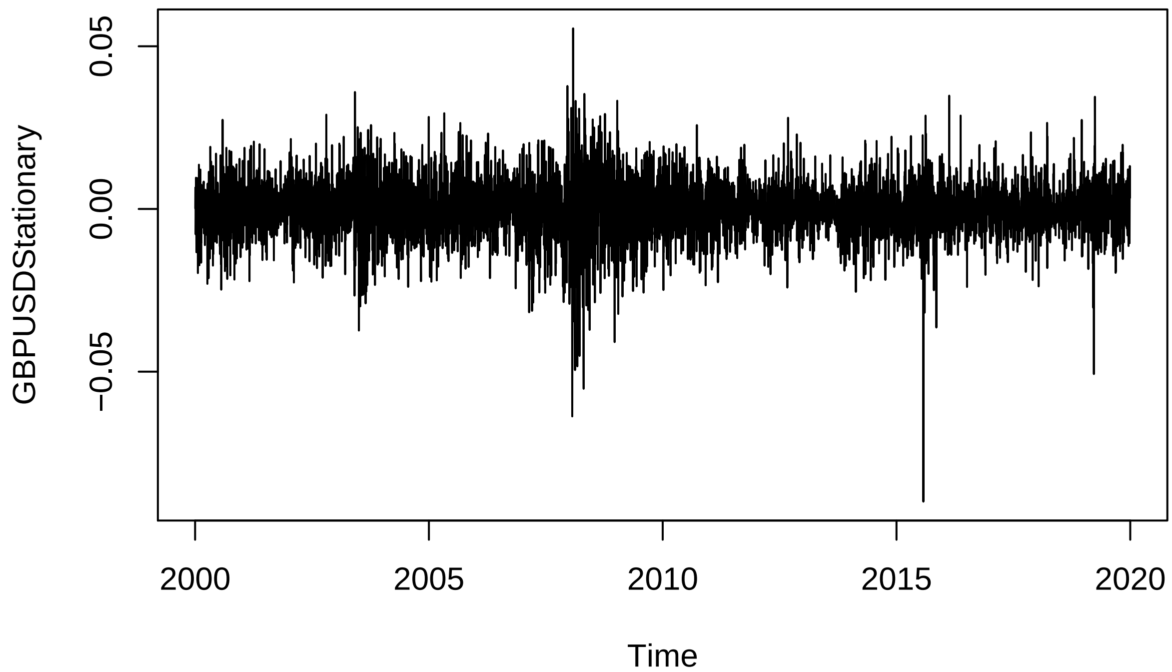
```
## Warning: package 'fUnitRoots' was built under R version 4.0.5

## 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(GBPUSDARIMATS, type = c("tau"), lags = c("short"),use.lag = NULL, doplot = TRUE)
```



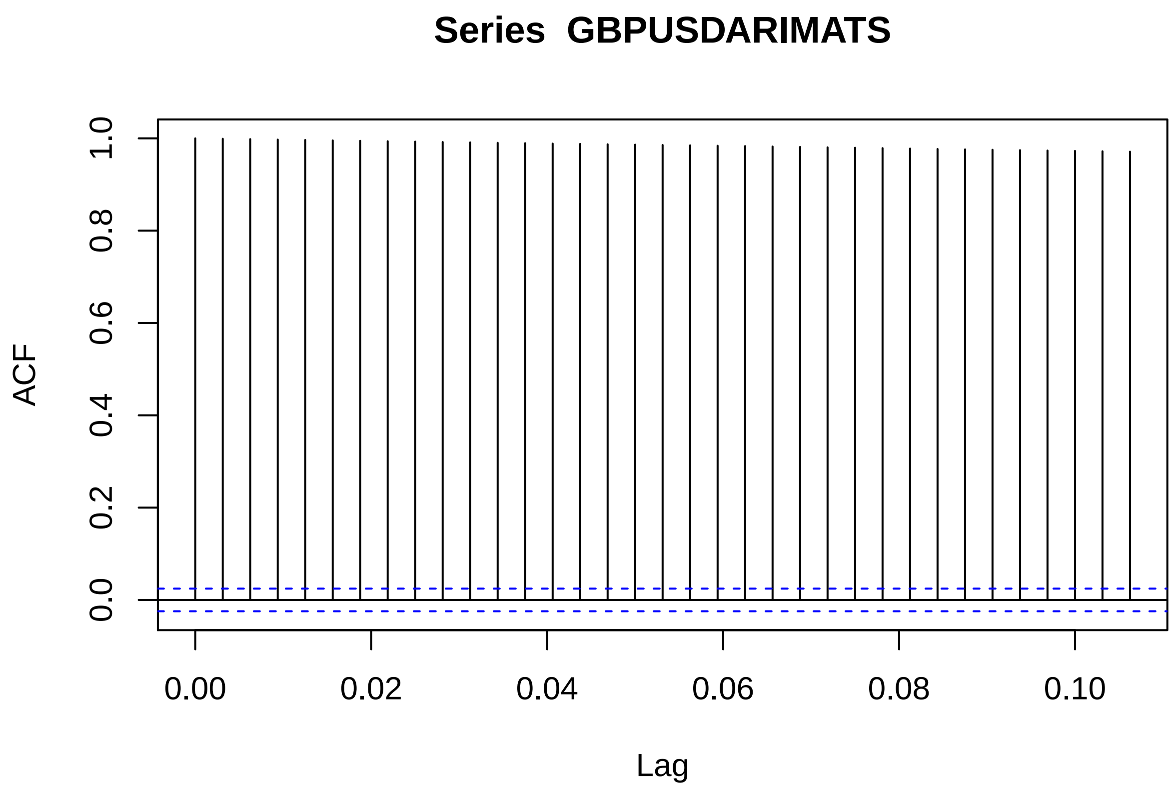

```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Tue May 04 00:29:15 2021 by user: janeo
```

```r
GBPUSDStationary= diff(GBPUSDARIMATS, differences=1)
plot(GBPUSDStationary)
```

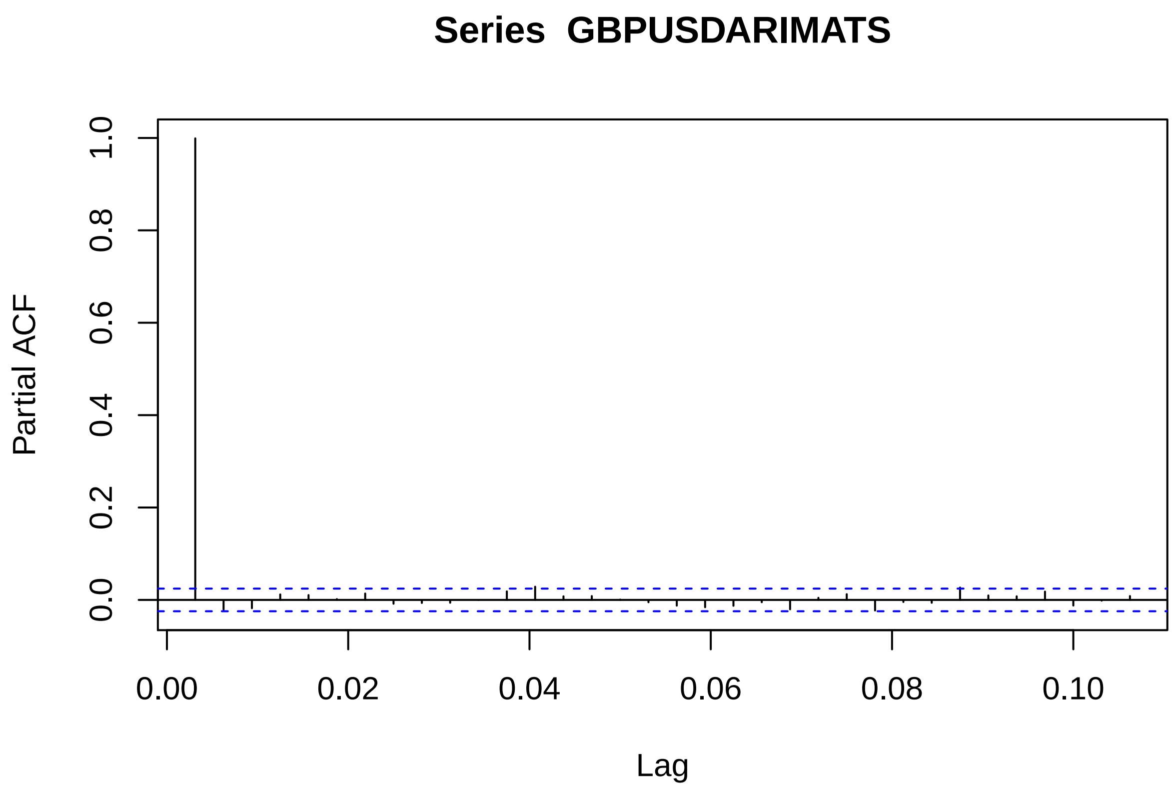


Calculating Autocorrlation function and partil autocorlation function

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

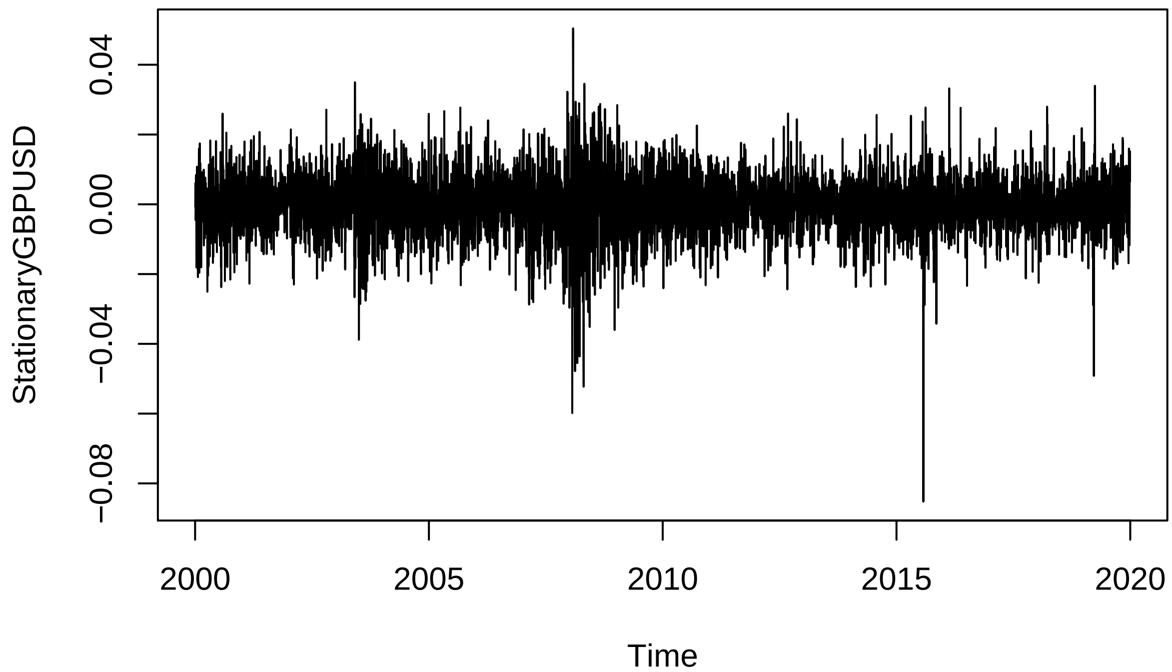



```
pacf(GBPUSDARIMATS, lag.max = 34)
```



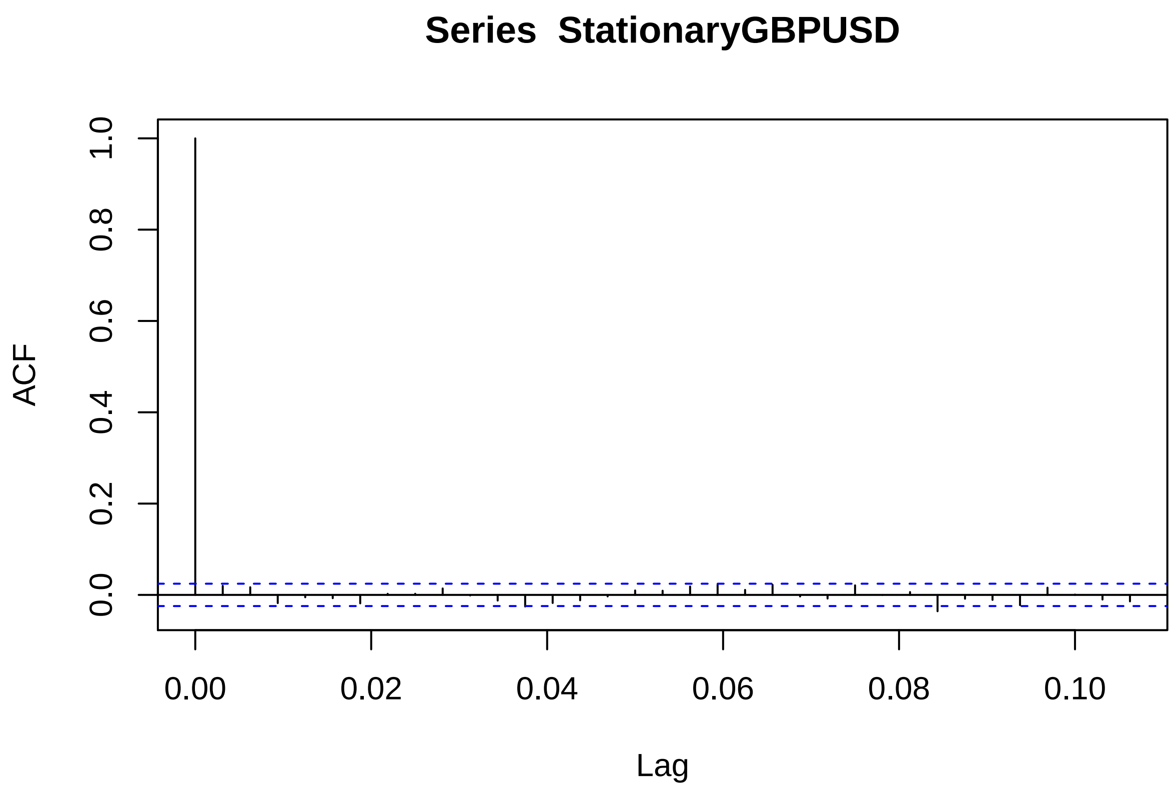
Adjusting and ensuring there are no seasonality

```
TSseasonallyadjustedGBPUSD <- GBPUSDARIMATS- ComponentGBPUSD$seasonal
StationaryGBPUSD <- diff(TSseasonallyadjustedGBPUSD, differences=1)
plot(StationaryGBPUSD)
```

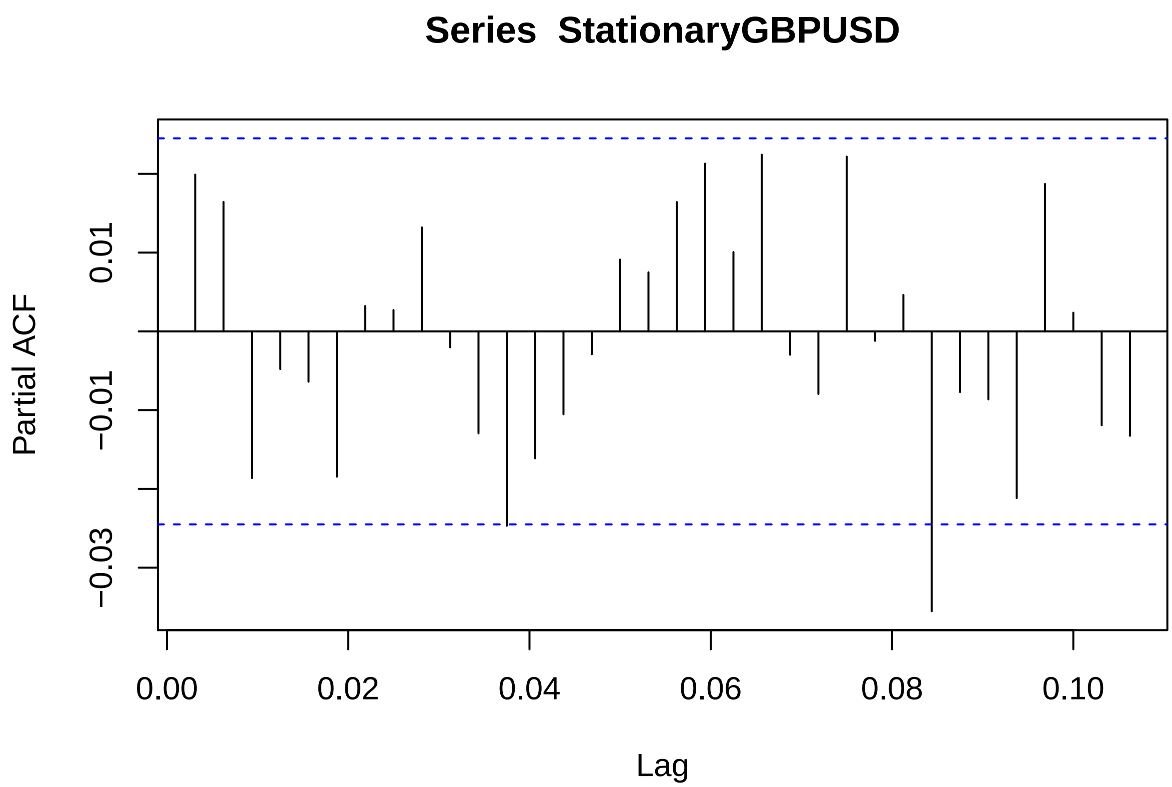


Calculating again for ACF and PACF after finding stationality

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




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



Fitting The ARIMA Model

ARIMA fitting (1,1,0)

```
fitArima1GBPUSD <- arima(GBPUSDARIMATS, order =  c(1,0,0), include.mean = TRUE)
fitArima1GBPUSD
```

```
##
## Call:
## arima(x = GBPUSDARIMATS, order = c(1, 0, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1  intercept
##       0.9993     1.5746
## s.e.  0.0005     0.1255
##
## sigma^2 estimated as 7.073e-05:  log likelihood = 21493.5,  aic = -42981
```

##Arima Fitting (0,1,0)

```
fitArima2GBPUSD <- arima(GBPUSDARIMATS, order =  c(0,1,0), include.mean = TRUE)
fitArima2GBPUSD
```

```
##
## Call:
## arima(x = GBPUSDARIMATS, order = c(0, 1, 0), include.mean = TRUE)
##
##
## sigma^2 estimated as 7.076e-05:  log likelihood = 21491.94,  aic = -42981.88
```

Arima Fitting (2,1,1)

```r
fitArima3GBPUSD <- arima(GBPUSDARIMATS, order = c(2,1,1), include.mean = TRUE)
fitArima3GBPUSD
```

```
##
## Call:
## arima(x = GBPUSDARIMATS, order = c(2, 1, 1), include.mean = TRUE)
##
## Coefficients:
##          ar1     ar2     ma1
##       0.0103  0.0128  0.0109
## s.e.  1.3905  0.0521  1.3902
##
## sigma^2 estimated as 7.072e-05:  log likelihood = 21493.89,  aic = -42979.79
```

##Fitting Arima (0,1,3)

```r
fitArima4GBPUSD <- arima(GBPUSDARIMATS, order = c(3,1,0), include.mean = TRUE)
fitArima4GBPUSD
```

```
##
## Call:
## arima(x = GBPUSDARIMATS, order = c(3, 1, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1     ar2      ar3
##       0.0212  0.0127  -0.0155
## s.e.  0.0125  0.0125   0.0125
##
## sigma^2 estimated as 7.07e-05:  log likelihood = 21494.65,  aic = -42981.3
```

##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
```

```
ARIMAModelSelectionGBPUSD = AIC(fitArima1GBPUSD,fitArima2GBPUSD,fitArima3GBPUSD,fitArima4GBPUSD)
```

```
## Warning in AIC.default(fitArima1GBPUSD, fitArima2GBPUSD, fitArima3GBPUSD, :
## models are not all fitted to the same number of observations
```

```
sortScore(ARIMAModelSelectionGBPUSD, score ="aic")
```

```
##                 df       AIC
## fitArima2GBPUSD  1 -42981.88
## fitArima4GBPUSD  4 -42981.30
## fitArima1GBPUSD  3 -42981.00
## fitArima3GBPUSD  4 -42979.79
```

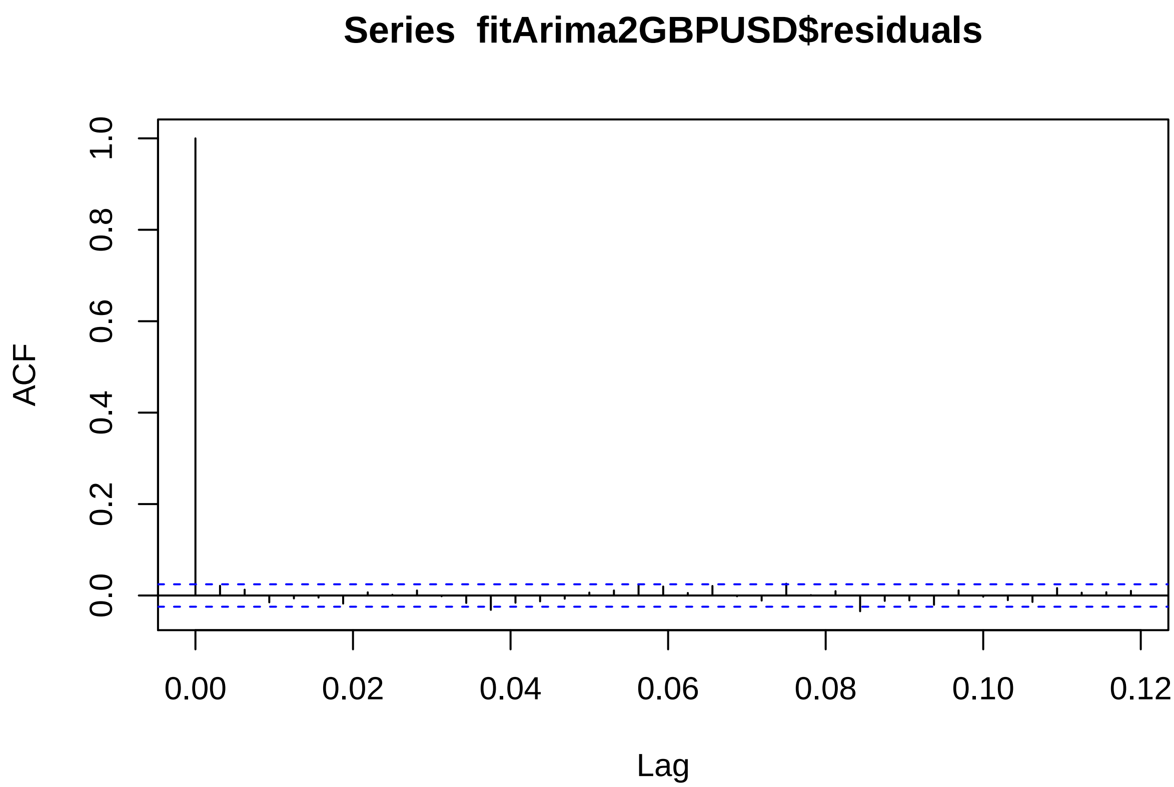
Base on the above the fitArima1CanJap is selected

```
confint(fitArima2GBPUSD)
```

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

Runing code to obtain Box Test Rest

```
acf(fitArima2GBPUSD$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
```

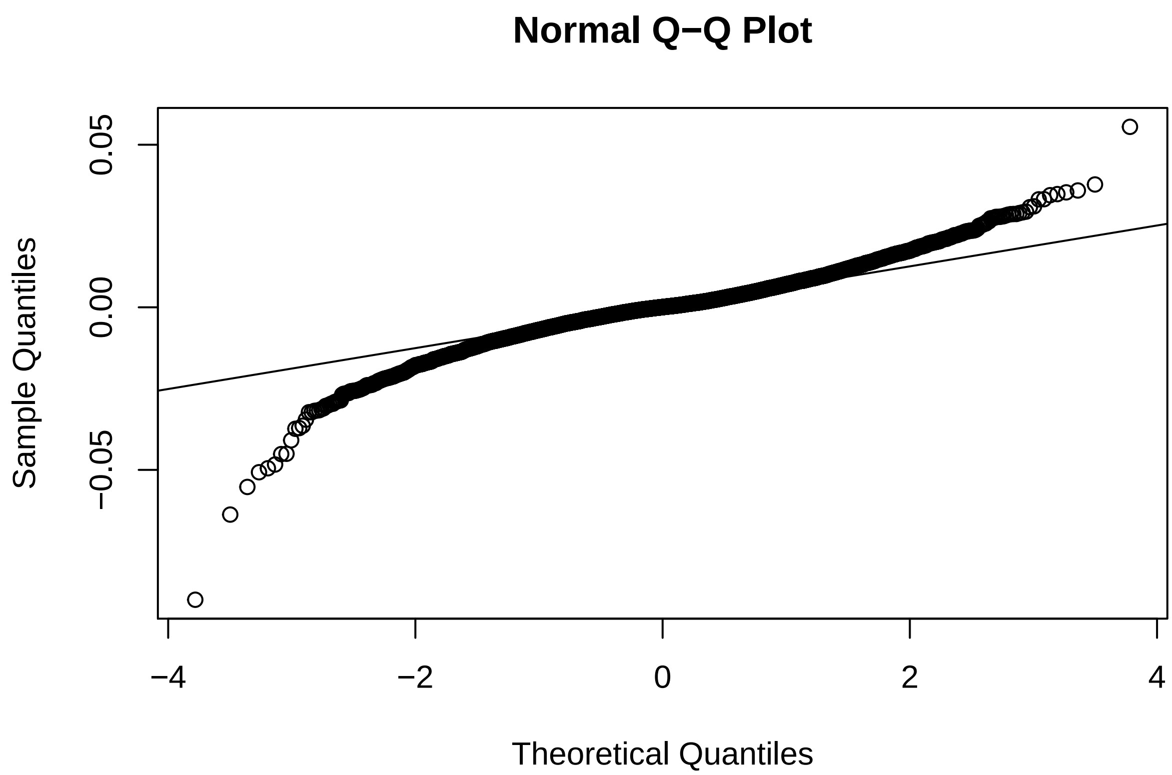
```
## Loading required package: bestglm
```

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

```r
library(bestglm)
 Box.test(resid(fitArima2GBPUSD),type="Ljung",lag=20,fitdf=1)
```

```
##
##  Box-Ljung test
##
## data:  resid(fitArima2GBPUSD)
## X-squared = 26.938, df = 19, p-value = 0.1061
```

```r
qqnorm(fitArima2GBPUSD$residuals)
qqline(fitArima2GBPUSD$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(GBPUSDARIMATS, trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[320] with drift         : Inf
##  ARIMA(0,1,0)             with drift         : -42970.48
##  ARIMA(1,1,0)(1,0,0)[320] with drift         : Inf
##  ARIMA(0,1,1)(0,0,1)[320] with drift         : -42969.78
##  ARIMA(0,1,0)                                : -42972.32
##  ARIMA(0,1,0)(1,0,0)[320] with drift         : Inf
##  ARIMA(0,1,0)(0,0,1)[320] with drift         : Inf
##  ARIMA(0,1,0)(1,0,1)[320] with drift         : Inf
##  ARIMA(1,1,0)             with drift         : -42970.38
##  ARIMA(0,1,1)             with drift         : -42971.3
##  ARIMA(1,1,1)             with drift         : -42968.34
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,1,0)                                : -42981.88
##
##  Best model: ARIMA(0,1,0)
```

```
## Series: GBPUSDARIMATS
## ARIMA(0,1,0)
##
## sigma^2 estimated as 7.076e-05:  log likelihood=21491.94
## AIC=-42981.88   AICc=-42981.88   BIC=-42975.11
```

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

```r
forecastarimaGBPUSD<- predict(fitArima2GBPUSD,n.ahead = 100)
forecastarimaGBPUSD
```

```
## $pred
## Time Series:
## Start = c(2019, 320)
## End = c(2020, 99)
## Frequency = 320
##   [1] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [11] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [21] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [31] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [41] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [51] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [61] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [71] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [81] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##  [91] 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663 1.3663
##
## $se
```

```
## Time Series:
## Start = c(2019, 320)
## End = c(2020, 99)
## Frequency = 320
##   [1] 0.008411936 0.011896274 0.014569900 0.016823872 0.018809660 0.020604951
##   [7] 0.022255890 0.023792548 0.025235808 0.026600877 0.027899235 0.029139801
##  [13] 0.030329666 0.031474582 0.032579287 0.033647743 0.034683300 0.035688821
##  [19] 0.036666778 0.037619321 0.038548333 0.039455476 0.040342227 0.041209901
##  [25] 0.042059679 0.042892625 0.043709701 0.044511781 0.045299661 0.046074070
##  [31] 0.046835677 0.047585095 0.048322892 0.049049593 0.049765684 0.050471615
##  [37] 0.051167808 0.051854655 0.052532523 0.053201754 0.053862670 0.054515575
##  [43] 0.055160752 0.055798470 0.056428981 0.057052525 0.057669327 0.058279601
##  [49] 0.058883551 0.059481369 0.060073238 0.060659332 0.061239817 0.061814852
##  [55] 0.062384586 0.062949164 0.063508723 0.064063395 0.064613305 0.065158575
##  [61] 0.065699319 0.066235649 0.066767671 0.067295487 0.067819195 0.068338890
##  [67] 0.068854662 0.069366600 0.069874787 0.070379305 0.070880231 0.071377643
##  [73] 0.071871611 0.072362208 0.072849501 0.073333557 0.073814438 0.074292206
##  [79] 0.074766921 0.075238642 0.075707423 0.076173319 0.076636383 0.077096666
##  [85] 0.077554217 0.078009084 0.078461314 0.078910953 0.079358044 0.079802630
##  [91] 0.080244754 0.080684454 0.081121772 0.081556744 0.081989409 0.082419802
##  [97] 0.082847960 0.083273916 0.083697705 0.084119358
```

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