ARIMA Model EUR and GBP

Jane

01/05/2021

Forcasting Exchange Rate Using ARIMA Model for EUR And GBP

Reading EUR and GBP Currency into r

```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
```

```r
EURGBPARIMA<-  read.csv ("EURGBP_Candlestick_1_D_BID_01.01.2000-31.12.2020.csv")%>%
  select('GMT.TIME', CLOSE)%>%
  rename(Date = ('GMT.TIME'), RateEURGBP = ("CLOSE"))


head(EURGBPARIMA)
```

```
##         Date RateEURGBP
## 1 2000-01-03     0.6261
## 2 2000-01-04     0.6293
## 3 2000-01-05     0.6281
## 4 2000-01-06     0.6263
## 5 2000-01-07     0.6277
## 6 2000-01-10     0.6264
```

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

```
##         Date RateEURGBP
## 1 2000-01-03     0.6261
## 2 2000-01-04     0.6293
## 3 2000-01-05     0.6281
## 4 2000-01-06     0.6263
## 5 2000-01-07     0.6277
## 6 2000-01-10     0.6264
```

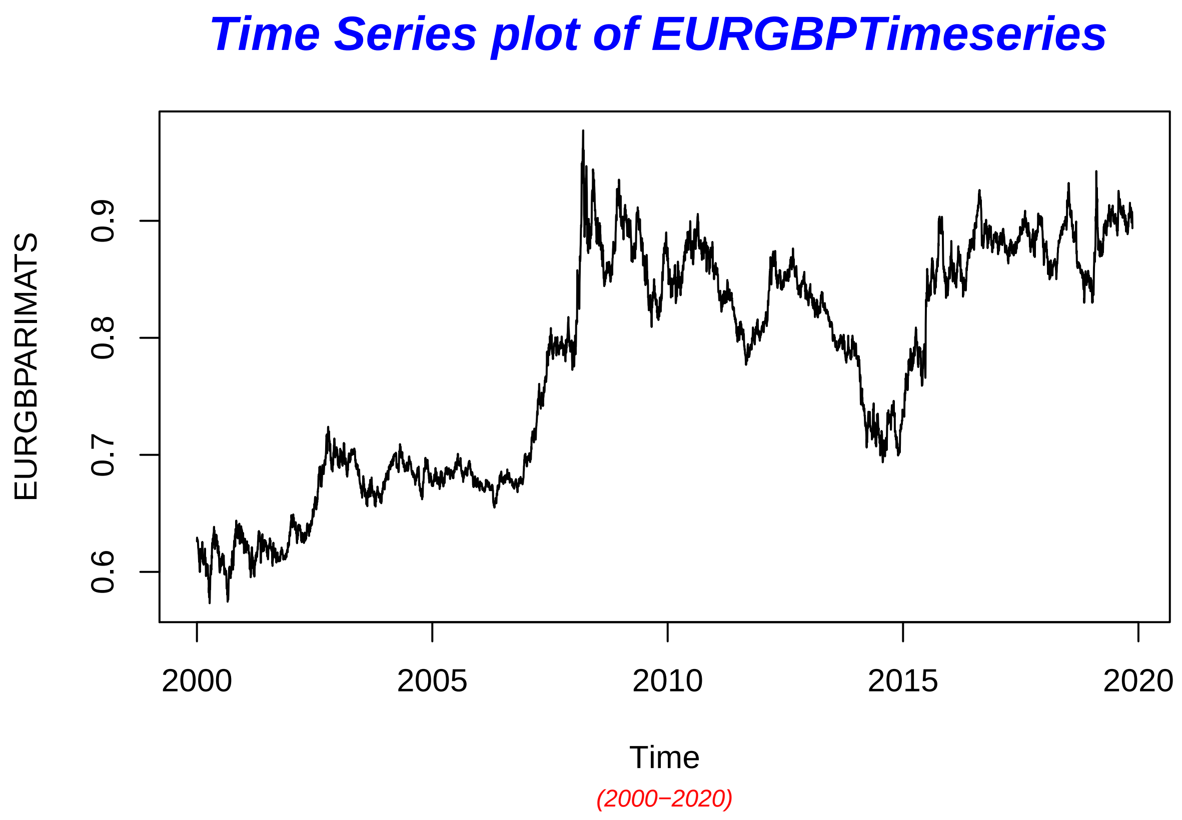
##Checking for obvious errors or missingg value

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

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

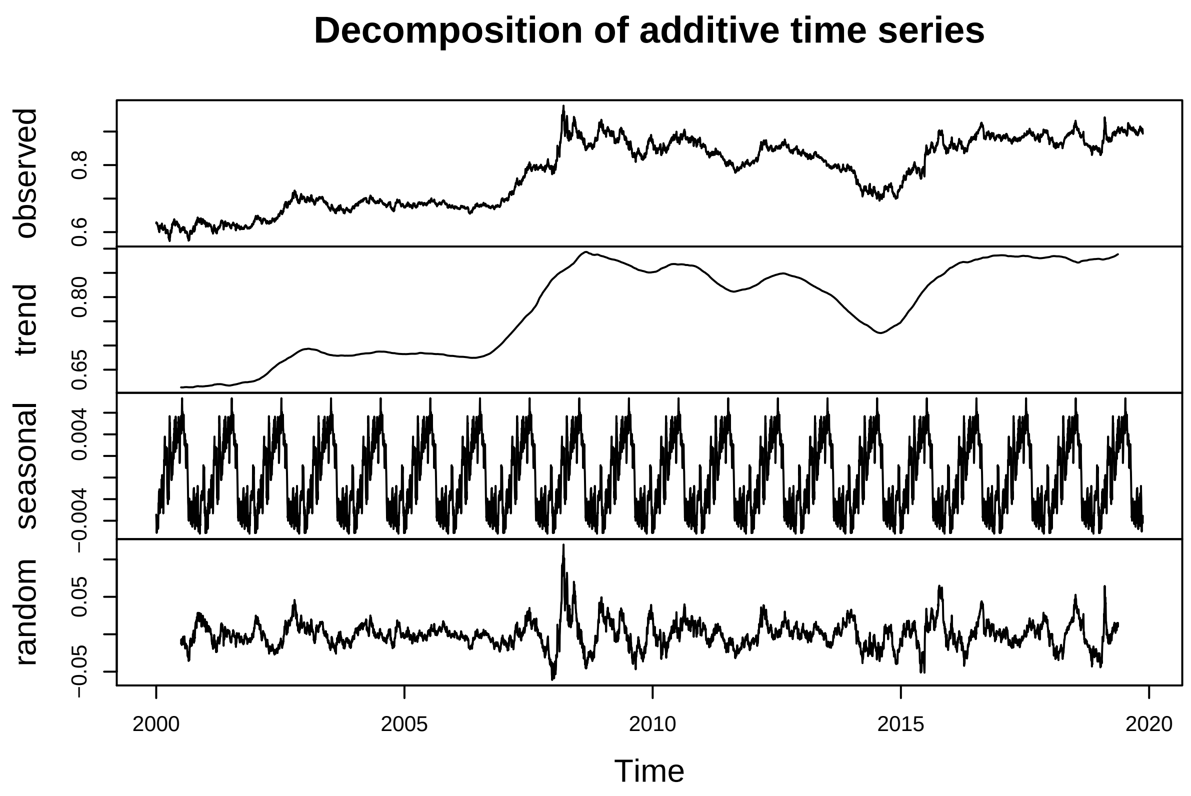
##Converting the data set into time series object

```r
#Converting the data set into time series object
EURGBPARIMATS<- ts(as.vector(EURGBPARIMA$Rate),  frequency = 322, start= c(2000,01,03))
plot.ts(EURGBPARIMATS)
title("Time Series plot of EURGBPTimeseries ", 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

```
ComponentEURGBP <- decompose(EURGBPARIMATS)
plot(ComponentEURGBP)
```



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
```

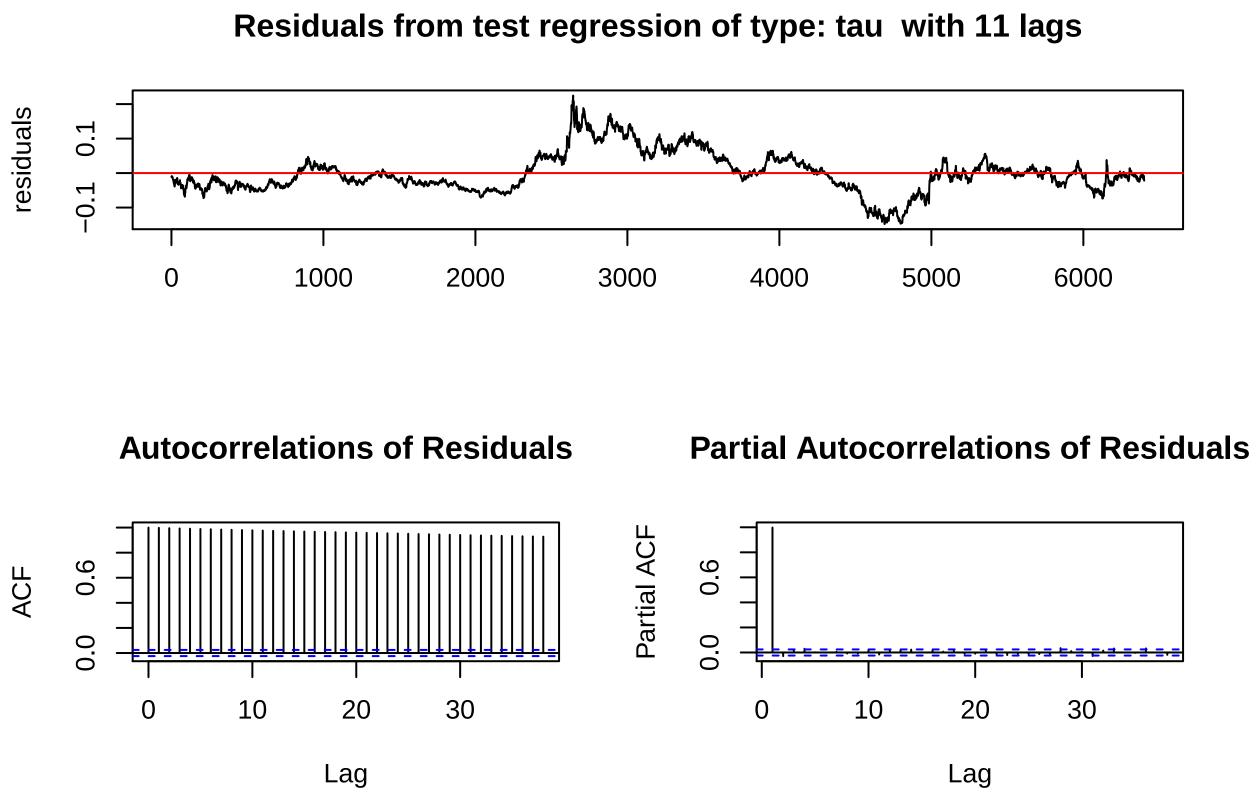
```
## Loading required package: timeSeries
```

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

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

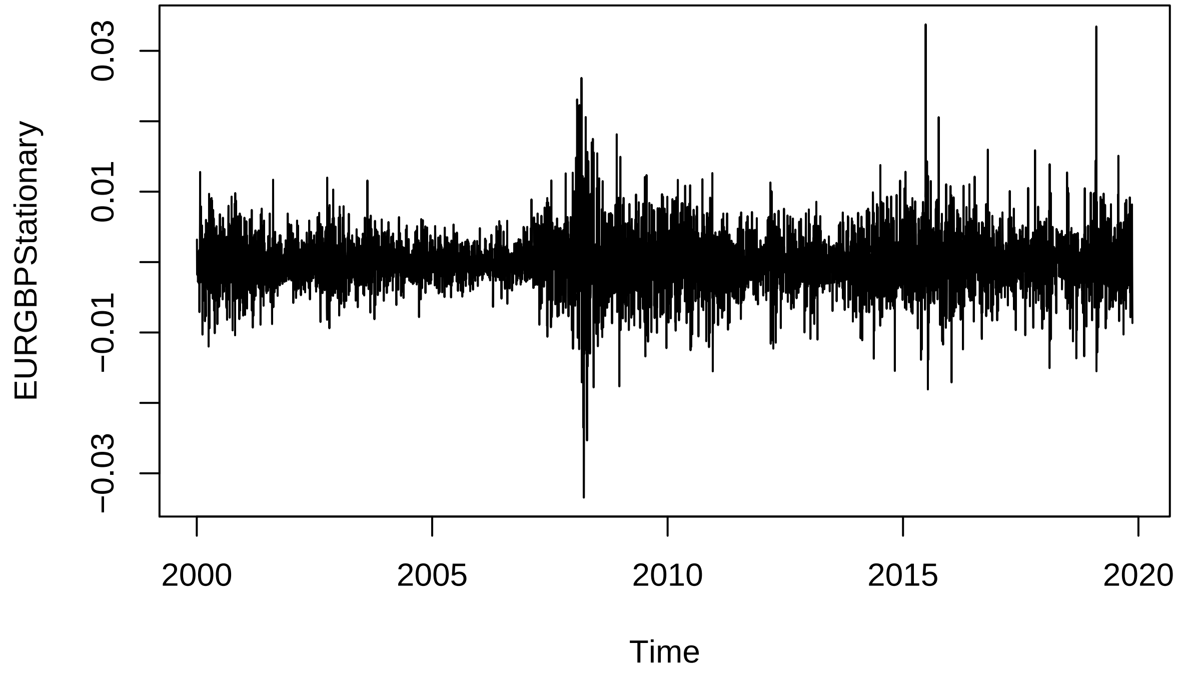
```
## Warning: package 'fBasics' was built under R version 4.0.5
```

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



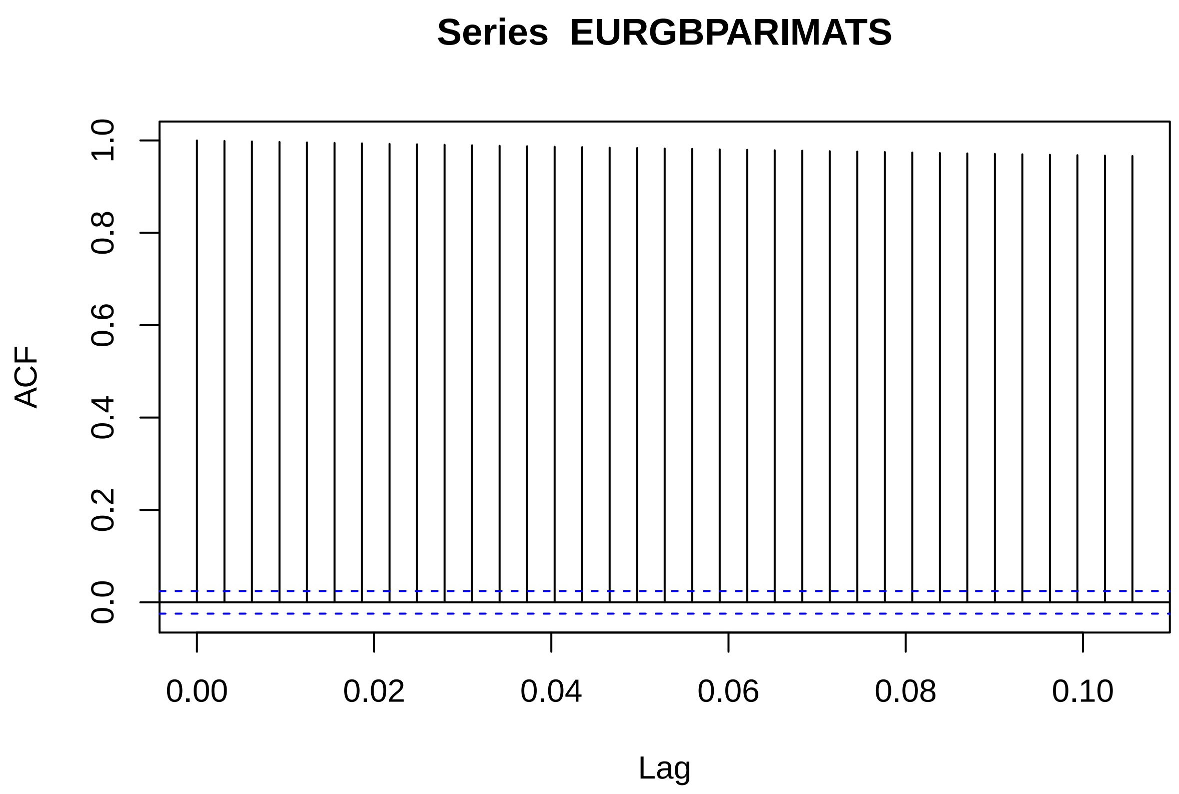
```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Tue May 04 00:04:46 2021 by user: janeo
```

```r
EURGBPStationary= diff(EURGBPARIMATS, differences=1)
plot(EURGBPStationary)
```

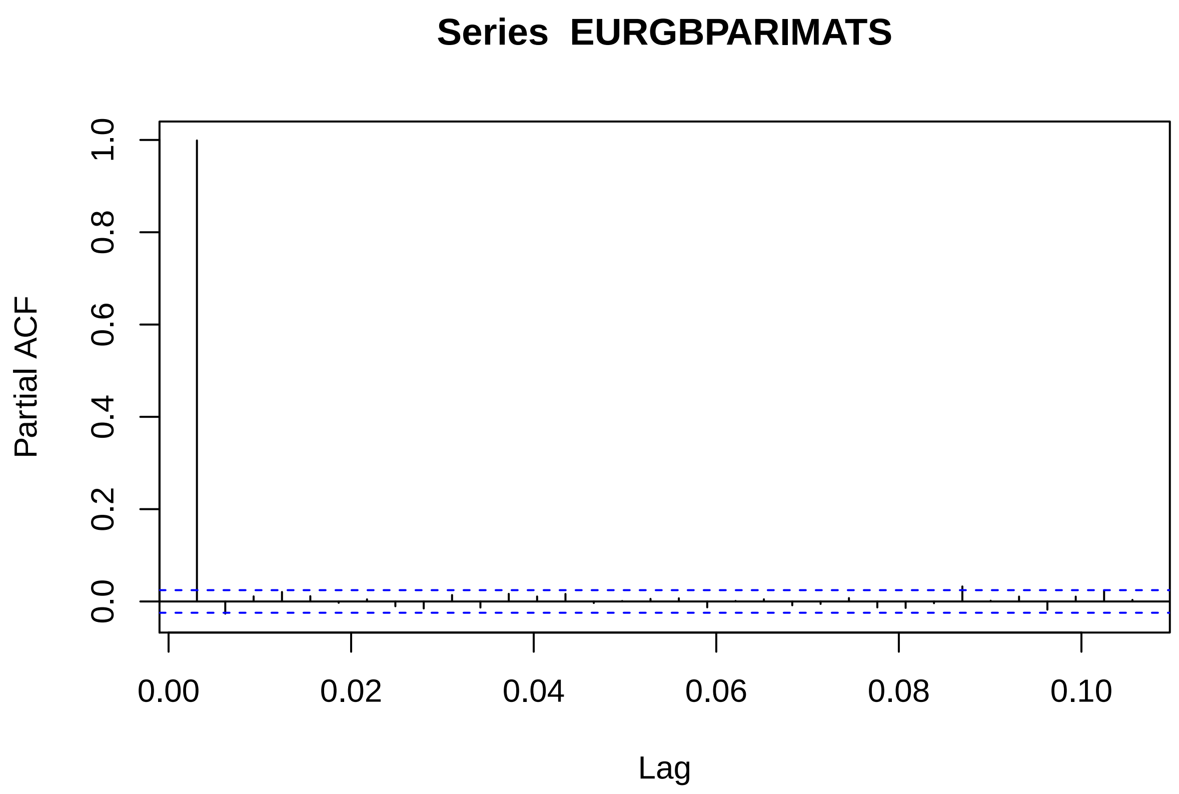


Calculating Autocorrlation function and partil autocorlation function

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

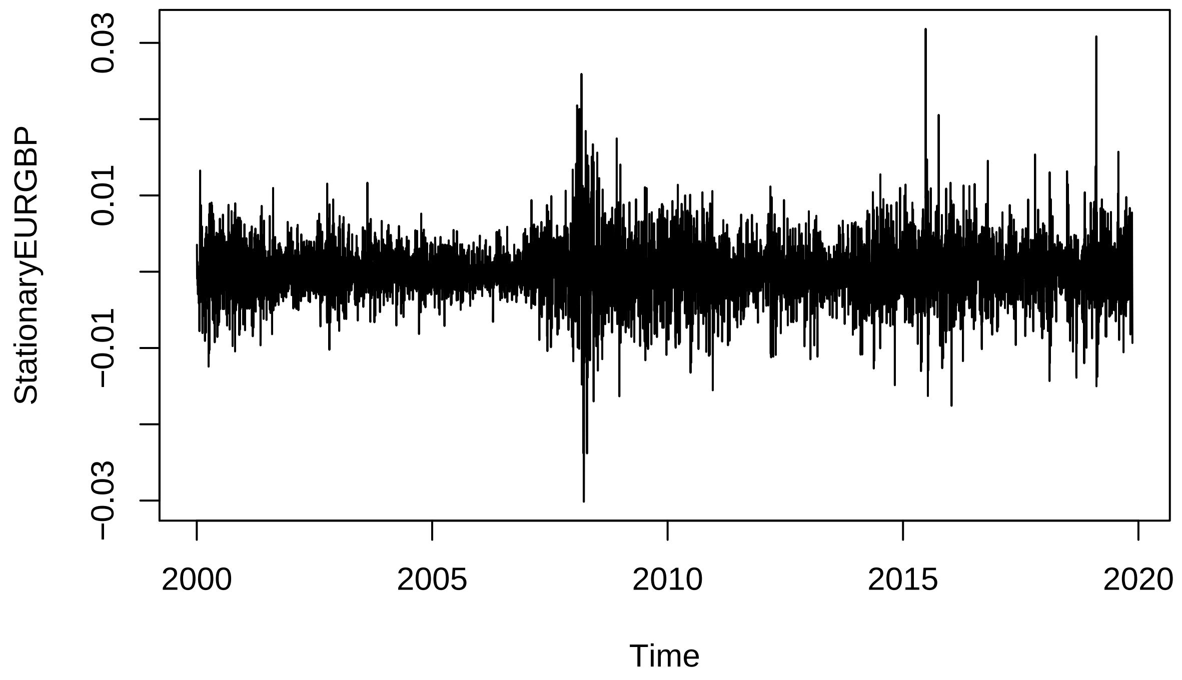


```r
pacf(EURGBPARIMATS, lag.max = 34)
```



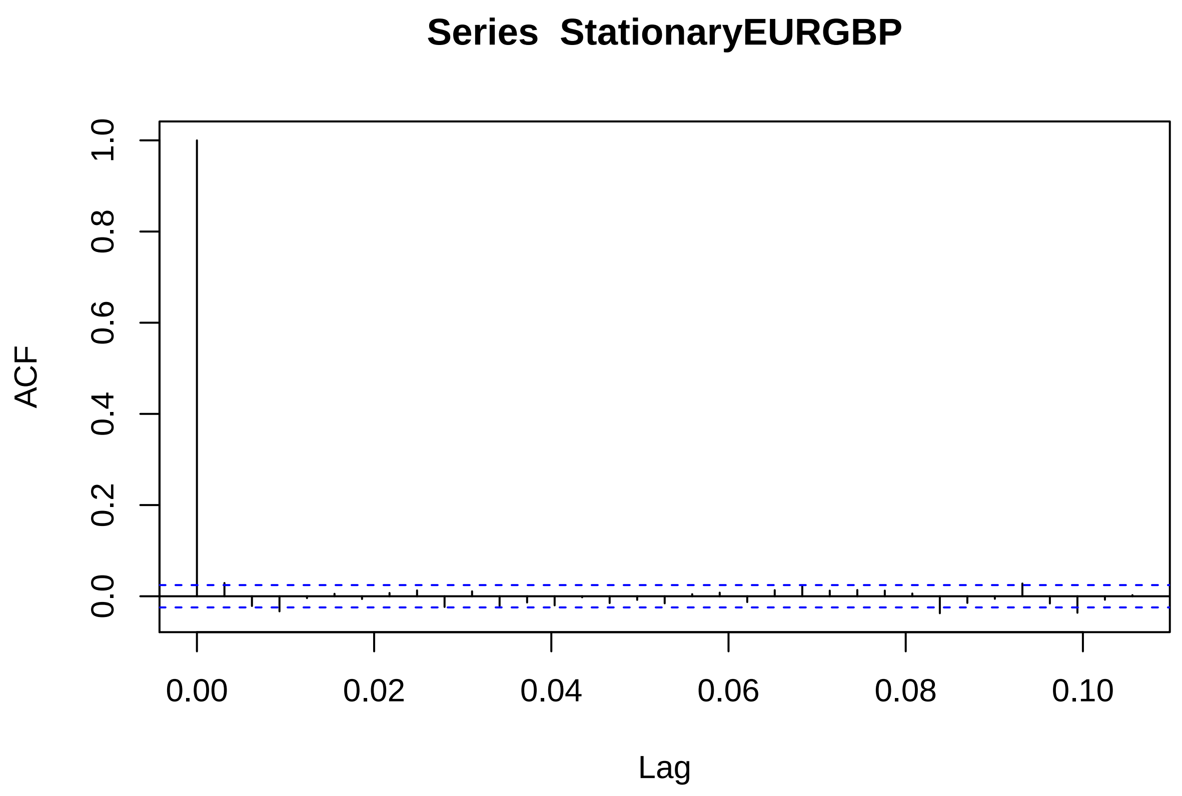
Adjusting and ensuring there are no seasonality

```
TSseasonallyadjustedEURGBP <- EURGBPARIMATS- ComponentEURGBP$seasonal
StationaryEURGBP <- diff(TSseasonallyadjustedEURGBP, differences=1)
plot(StationaryEURGBP)
```

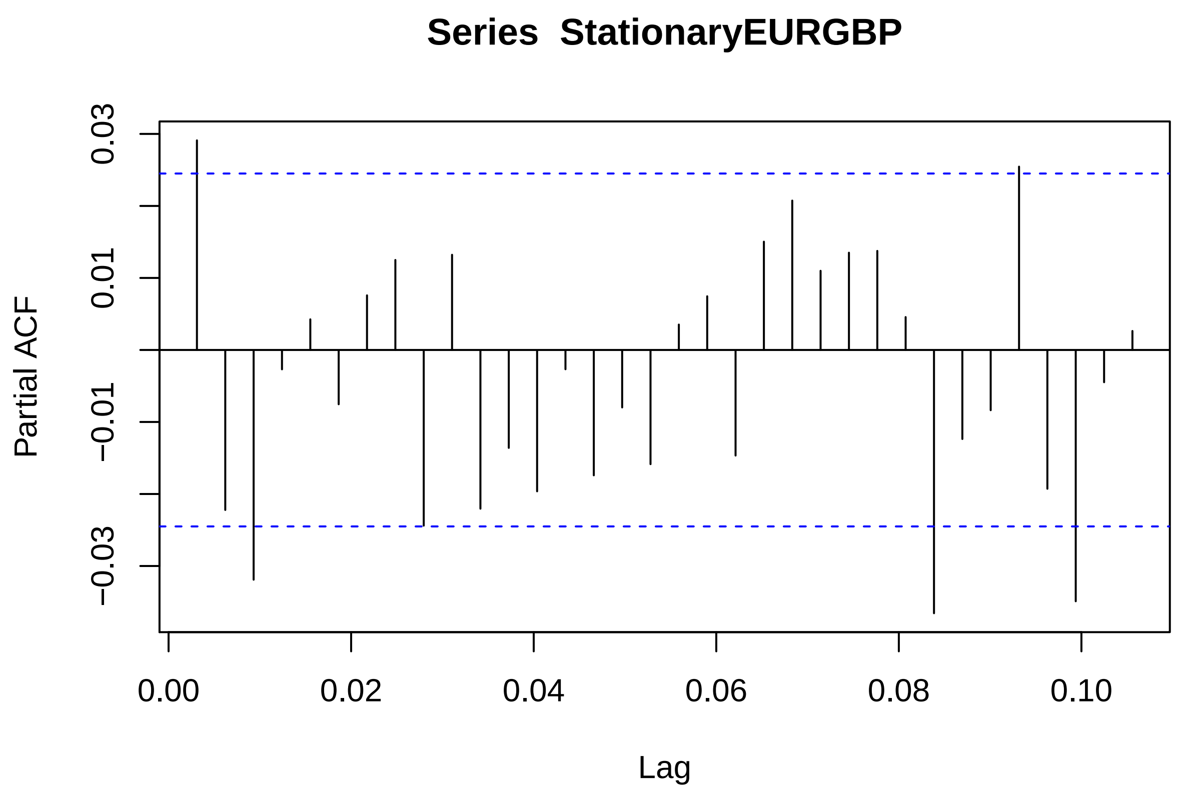


Calculating again for ACF and PACF after finding stationality

```r
acf(StationaryEURGBP, lag.max=34)
```



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



Fitting The ARIMA Model

ARIMA fitting (1,1,0)

```
fitArima1EURGBP <- arima(EURGBPARIMATS, order =  c(1,0,0), include.mean = TRUE)
fitArima1EURGBP
```

```
##
## Call:
## arima(x = EURGBPARIMATS, order = c(1, 0, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1  intercept
##       0.9992     0.7768
## s.e.  0.0004     0.0492
##
## sigma^2 estimated as 1.486e-05:  log likelihood = 26490.05,  aic = -52974.11
```

##Arima Fitting (0,1,0)

```
fitArima2EURGBP <- arima(EURGBPARIMATS, order =  c(0,1,0), include.mean = TRUE)
fitArima2EURGBP
```

```
##
## Call:
## arima(x = EURGBPARIMATS, order = c(0, 1, 0), include.mean = TRUE)
##
##
## sigma^2 estimated as 1.486e-05:  log likelihood = 26488.42,  aic = -52974.84
```

Arima Fitting (2,1,1)

```
fitArima3EURGBP <- arima(EURGBPARIMATS, order = c(2,1,1), include.mean = TRUE)
fitArima3EURGBP
```

```
##
## Call:
## arima(x = EURGBPARIMATS, order = c(2, 1, 1), include.mean = TRUE)
##
## Coefficients:
##          ar1      ar2      ma1
##       0.6699  -0.0472  -0.6397
## s.e.  0.3385   0.0125   0.3393
##
## sigma^2 estimated as 1.483e-05:  log likelihood = 26495.56,  aic = -52983.12
```

##Fitting Arima (3,1,0)

```
fitArima4EURGBP <- arima(EURGBPARIMATS, order = c(3,1,0), include.mean = TRUE)
fitArima4EURGBP
```

```
##
## Call:
## arima(x = EURGBPARIMATS, order = c(3, 1, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1      ar2      ar3
##       0.0301  -0.0237  -0.0311
## s.e.  0.0125   0.0125   0.0125
##
## sigma^2 estimated as 1.482e-05:  log likelihood = 26496.36,  aic = -52984.73
```

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

```
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
```

```
ARIMAModelSelectionEURGBP = AIC(fitArima1EURGBP,fitArima2EURGBP,fitArima3EURGBP,fitArima4EURGBP)
```

```
## Warning in AIC.default(fitArima1EURGBP, fitArima2EURGBP, fitArima3EURGBP, :
## models are not all fitted to the same number of observations
```

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

```
##                 df       AIC
## fitArima4EURGBP  4 -52984.73
## fitArima3EURGBP  4 -52983.12
## fitArima2EURGBP  1 -52974.84
## fitArima1EURGBP  3 -52974.11
```

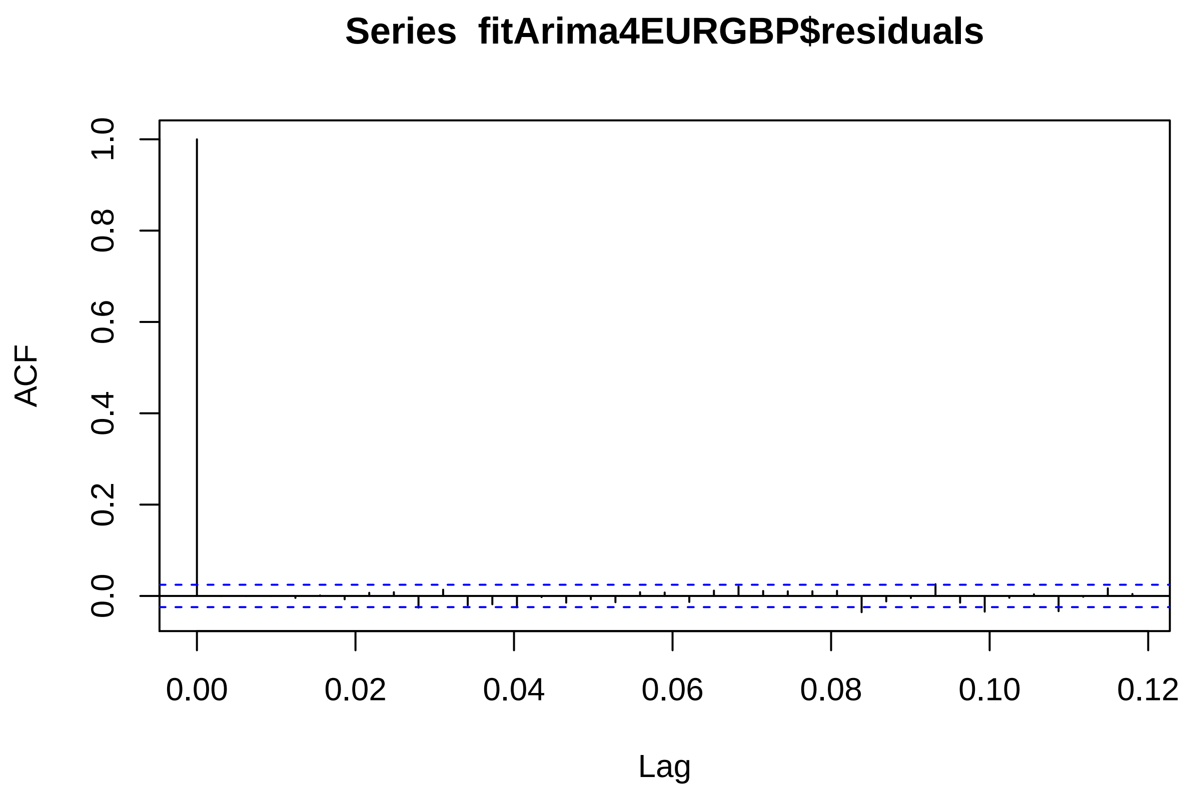
Base on the above the fitArima1CanJap is selected

```
confint(fitArima4EURGBP)
```

```
##             2.5 %       97.5 %
## ar1  0.005629195  0.054624593
## ar2 -0.048158234  0.000851632
## ar3 -0.055621483 -0.006622301
```

Runing code to obtain Box Test Rest

```r
acf(fitArima4EURGBP$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
```

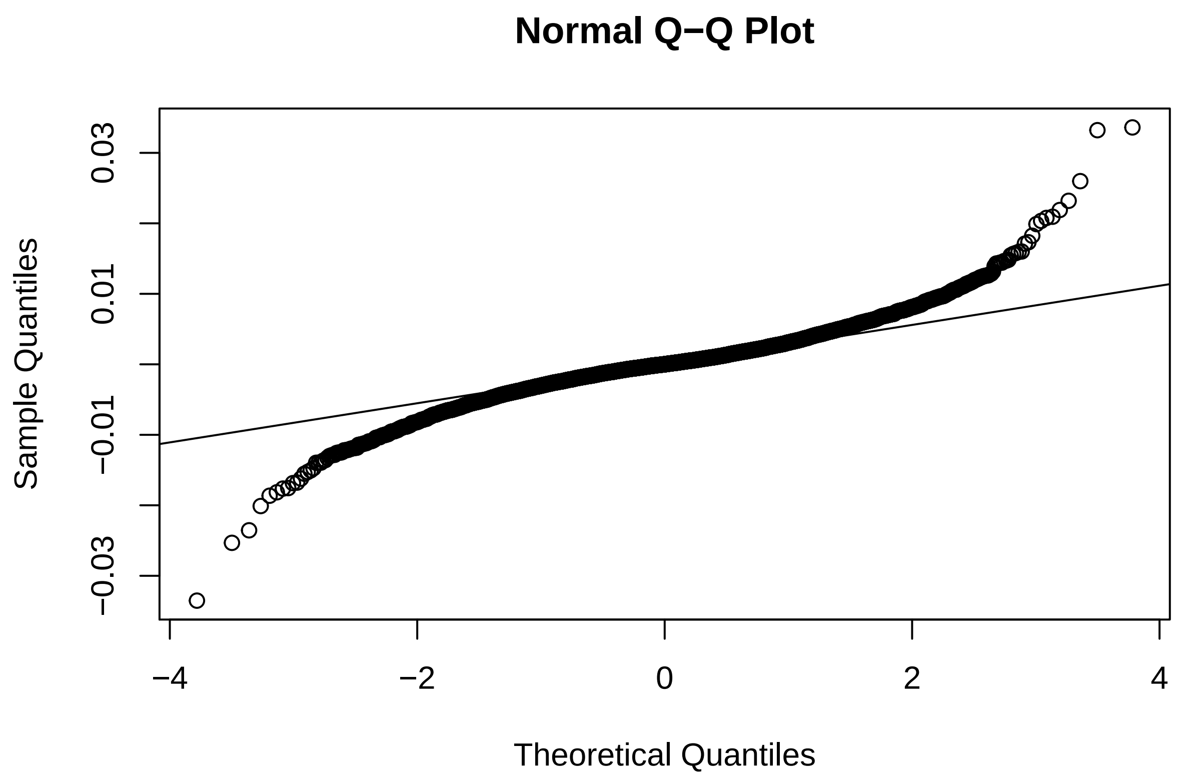
```
## Loading required package: bestglm
```

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

```r
library(bestglm)
 Box.test(resid(fitArima4EURGBP),type="Ljung",lag=20,fitdf=1)
```

```
##
##  Box-Ljung test
##
## data:  resid(fitArima4EURGBP)
## X-squared = 20.357, df = 19, p-value = 0.3734
```

```
qqnorm(fitArima4EURGBP$residuals)
qqline(fitArima4EURGBP$residuals)
```



Using Auto.arima to find the best model fit

```
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(EURGBPARIMATS, trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[322] with drift         : Inf
##  ARIMA(0,1,0)             with drift         : -52962.48
##  ARIMA(1,1,0)(1,0,0)[322] with drift         : Inf
##  ARIMA(0,1,1)(0,0,1)[322] with drift         : Inf
##  ARIMA(0,1,0)                                : -52963.72
##  ARIMA(0,1,0)(1,0,0)[322] with drift         : Inf
##  ARIMA(0,1,0)(0,0,1)[322] with drift         : -52960.97
##  ARIMA(0,1,0)(1,0,1)[322] with drift         : Inf
##  ARIMA(1,1,0)             with drift         : -52965.92
##  ARIMA(1,1,0)(0,0,1)[322] with drift         : Inf
##  ARIMA(1,1,0)(1,0,1)[322] with drift         : Inf
##  ARIMA(2,1,0)             with drift         : -52966.95
##  ARIMA(2,1,0)(1,0,0)[322] with drift         : Inf
##  ARIMA(2,1,0)(0,0,1)[322] with drift         : Inf
##  ARIMA(2,1,0)(1,0,1)[322] with drift         : Inf
##  ARIMA(3,1,0)             with drift         : -52970.39
##  ARIMA(3,1,0)(1,0,0)[322] with drift         : Inf
##  ARIMA(3,1,0)(0,0,1)[322] with drift         : Inf
##  ARIMA(3,1,0)(1,0,1)[322] with drift         : Inf
##  ARIMA(4,1,0)             with drift         : -52967.67
##  ARIMA(3,1,1)             with drift         : -52968.37
##  ARIMA(2,1,1)             with drift         : -52969.72
##  ARIMA(4,1,1)             with drift         : Inf
##  ARIMA(3,1,0)                                : -52971.6
##  ARIMA(3,1,0)(1,0,0)[322]                    : Inf
##  ARIMA(3,1,0)(0,0,1)[322]                    : Inf
##  ARIMA(3,1,0)(1,0,1)[322]                    : Inf
##  ARIMA(2,1,0)                                : -52968.22
##  ARIMA(4,1,0)                                : -52968.88
##  ARIMA(3,1,1)                                : -52969.58
##  ARIMA(2,1,1)                                : -52970.95
##  ARIMA(4,1,1)                                : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,1,0)                                : -52984.72
##
##  Best model: ARIMA(3,1,0)
```

```
## Series: EURGBPARIMATS
## ARIMA(3,1,0)
##
## Coefficients:
##          ar1      ar2      ar3
##       0.0301  -0.0237  -0.0311
## s.e.  0.0125   0.0125   0.0125
##
## sigma^2 estimated as 1.483e-05:  log likelihood=26496.36
## AIC=-52984.73   AICc=-52984.72   BIC=-52957.67
```

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

```
forecastarimaEURGBP<- predict(fitArima4EURGBP,n.ahead = 100)
forecastarimaEURGBP
```

```
## $pred
## Time Series:
## Start = c(2019, 283)
## End = c(2020, 60)
## Frequency = 322
##   [1] 0.8935096 0.8938773 0.8941612 0.8941655 0.8941474 0.8941379 0.8941380
##   [8] 0.8941387 0.8941391 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [15] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [22] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [29] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [36] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [43] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [50] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [57] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [64] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [71] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [78] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [85] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [92] 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390 0.8941390
##  [99] 0.8941390 0.8941390
##
## $se
## Time Series:
## Start = c(2019, 283)
## End = c(2020, 60)
## Frequency = 322
##   [1] 0.003850151 0.005527564 0.006752576 0.007725608 0.008586788 0.009371355
##   [7] 0.010096689 0.010773355 0.011409873 0.012012667 0.012586625 0.013135530
##  [13] 0.013662402 0.014169697 0.014659446 0.015133355 0.015592867 0.016039220
##  [19] 0.016473483 0.016896588 0.017309355 0.017712505 0.018106681 0.018492457
##  [25] 0.018870348 0.019240819 0.019604289 0.019961143 0.020311728 0.020656364
##  [31] 0.020995343 0.021328936 0.021657391 0.021980938 0.022299791 0.022614150
##  [37] 0.022924197 0.023230108 0.023532041 0.023830150 0.024124575 0.024415449
##  [43] 0.024702899 0.024987043 0.025267991 0.025545850 0.025820718 0.026092692
##  [49] 0.026361859 0.026628306 0.026892113 0.027153358 0.027412112 0.027668447
##  [55] 0.027922429 0.028174121 0.028423585 0.028670878 0.028916056 0.029159173
##  [61] 0.029400279 0.029639425 0.029876656 0.030112018 0.030345555 0.030577308
##  [67] 0.030807318 0.031035623 0.031262261 0.031487268 0.031710678 0.031932525
##  [73] 0.032152842 0.032371659 0.032589007 0.032804915 0.033019411 0.033232523
##  [79] 0.033444276 0.033654698 0.033863812 0.034071642 0.034278213 0.034483546
##  [85] 0.034687664 0.034890587 0.035092338 0.035292934 0.035492398 0.035690746
##  [91] 0.035887998 0.036084172 0.036279285 0.036473355 0.036666397 0.036858428
##  [97] 0.037049464 0.037239520 0.037428611 0.037616752
```

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