ARIMA Model EUR And JPY

Jane

29/04/2021

# Forcasting Exchange Rate Using ARIMA Model for EUR And US Dollar

Reading EUR and EUR 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
```

```
EURUSDARIMA <-  read.csv ("EURUSD_Candlestick_1_D_BID_01.01.2000-31.12.2020.csv")%>%
  select(’GMT.TIME’, CLOSE)%>%
  rename(Date = (’GMT.TIME’), RateEURUSD = ("CLOSE"))


head(EURUSDARIMA)
```

```
##         Date RateEURUSD
## 1 2000-01-03     1.0243
## 2 2000-01-04     1.0295
## 3 2000-01-05     1.0321
## 4 2000-01-06     1.0324
## 5 2000-01-07     1.0296
## 6 2000-01-10     1.0253
```

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

```
##         Date RateEURUSD
## 1 2000-01-03     1.0243
## 2 2000-01-04     1.0295
## 3 2000-01-05     1.0321
## 4 2000-01-06     1.0324
## 5 2000-01-07     1.0296
## 6 2000-01-10     1.0253
```

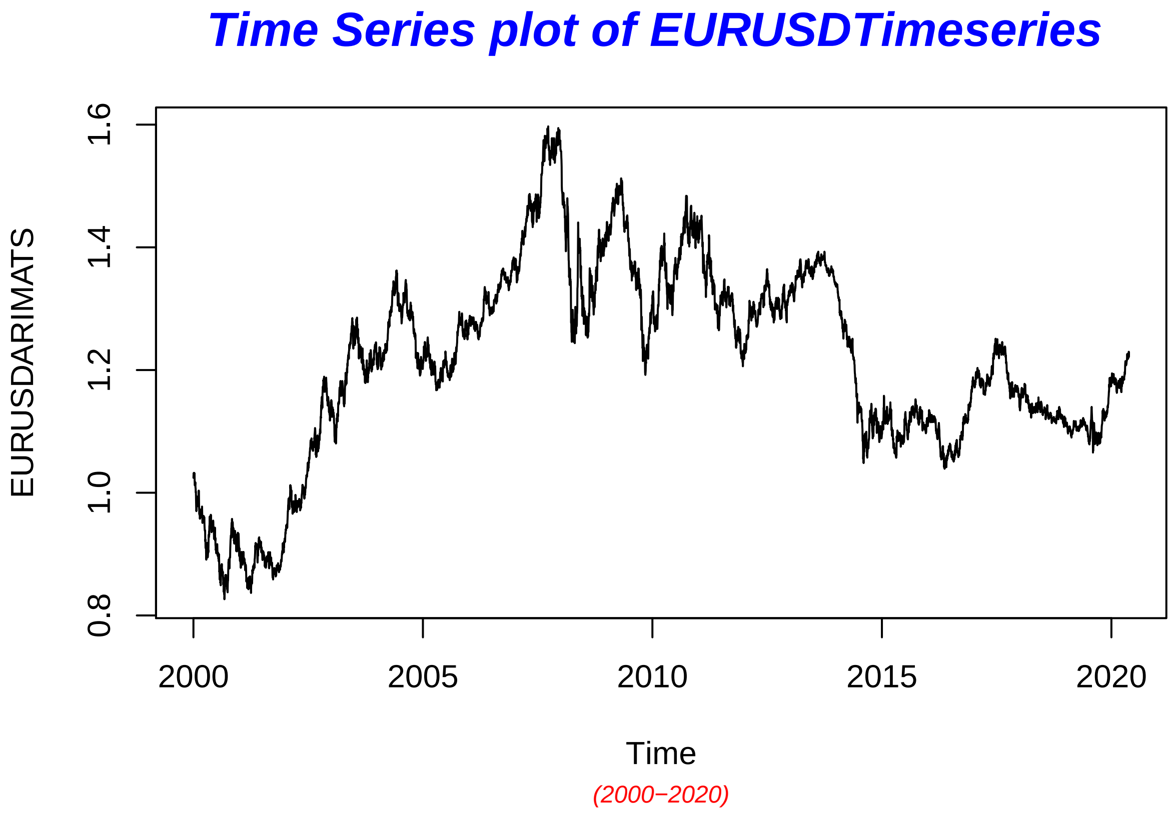
##Checking for obvious errors or missingg value

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

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

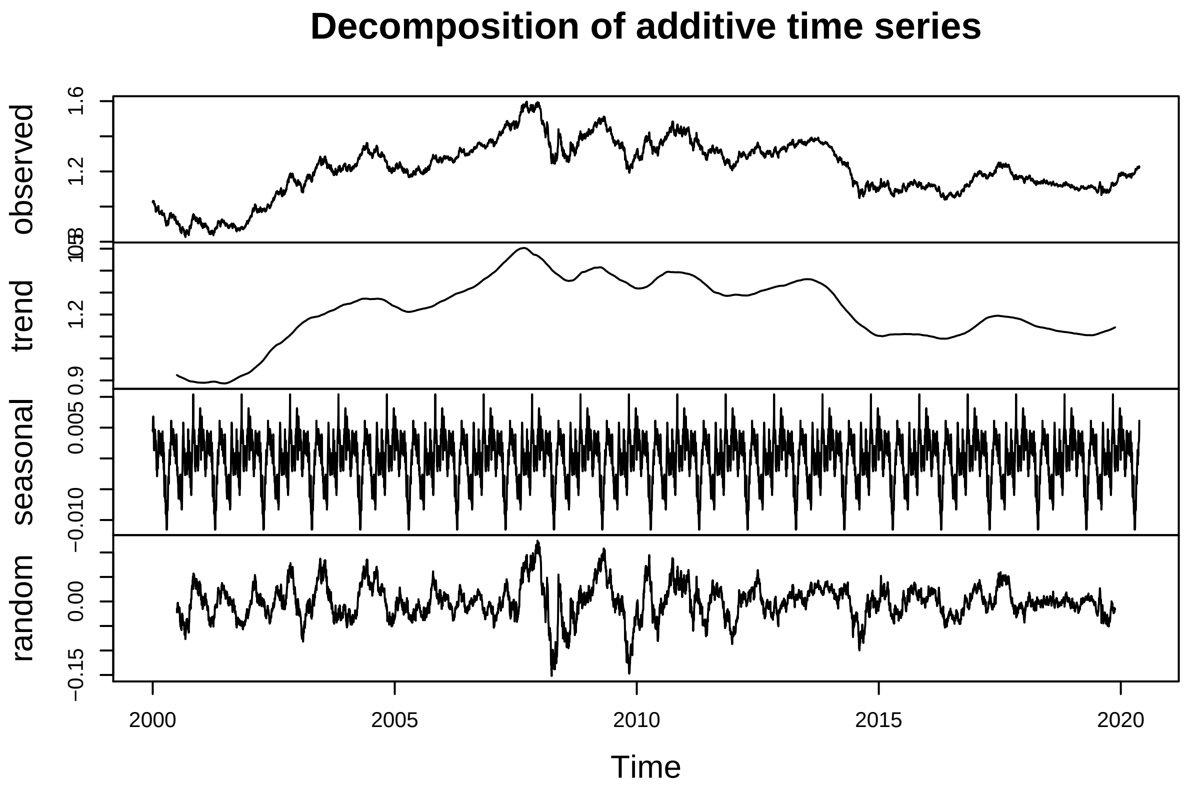
##Converting the data set into time series object

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

```r
ComponentEURUSD <- decompose(EURUSDARIMATS)
plot(ComponentEURUSD)
```

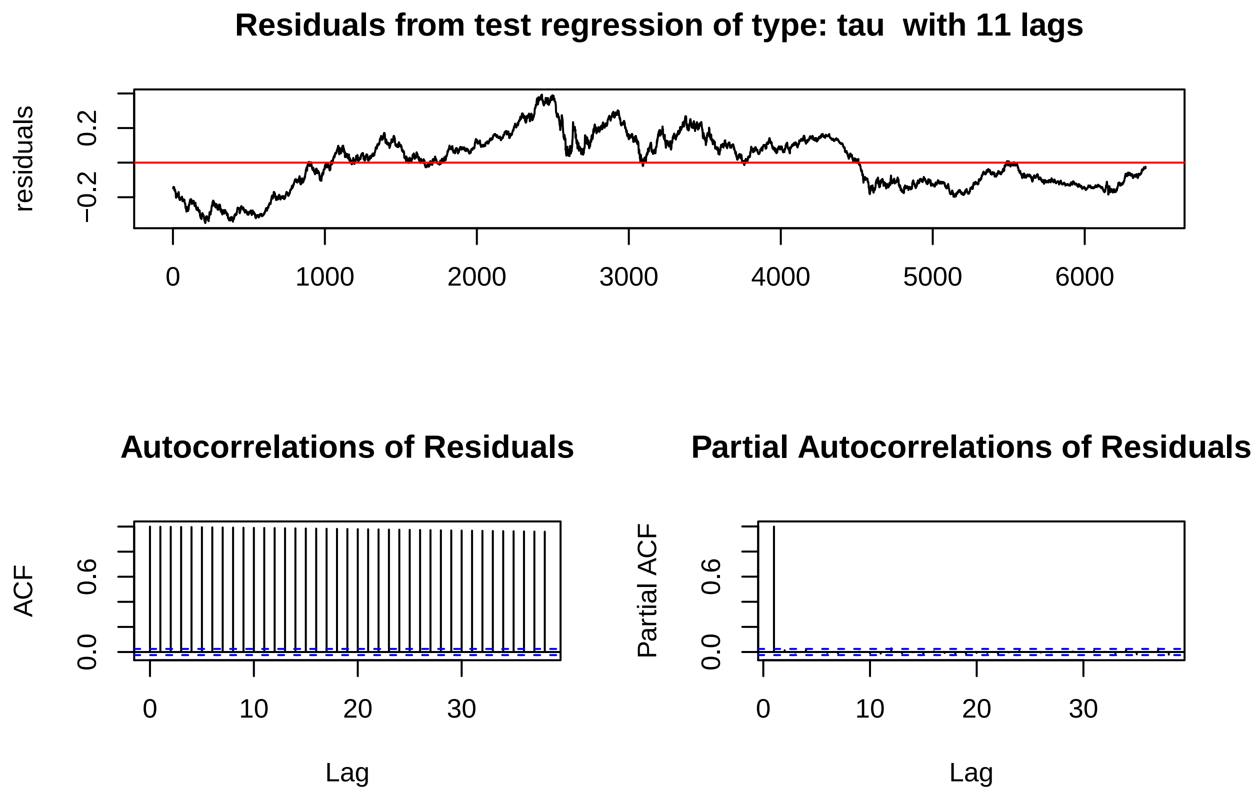


To To achieve stationarity by differencing the data – compute the differences between consecutive observations

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

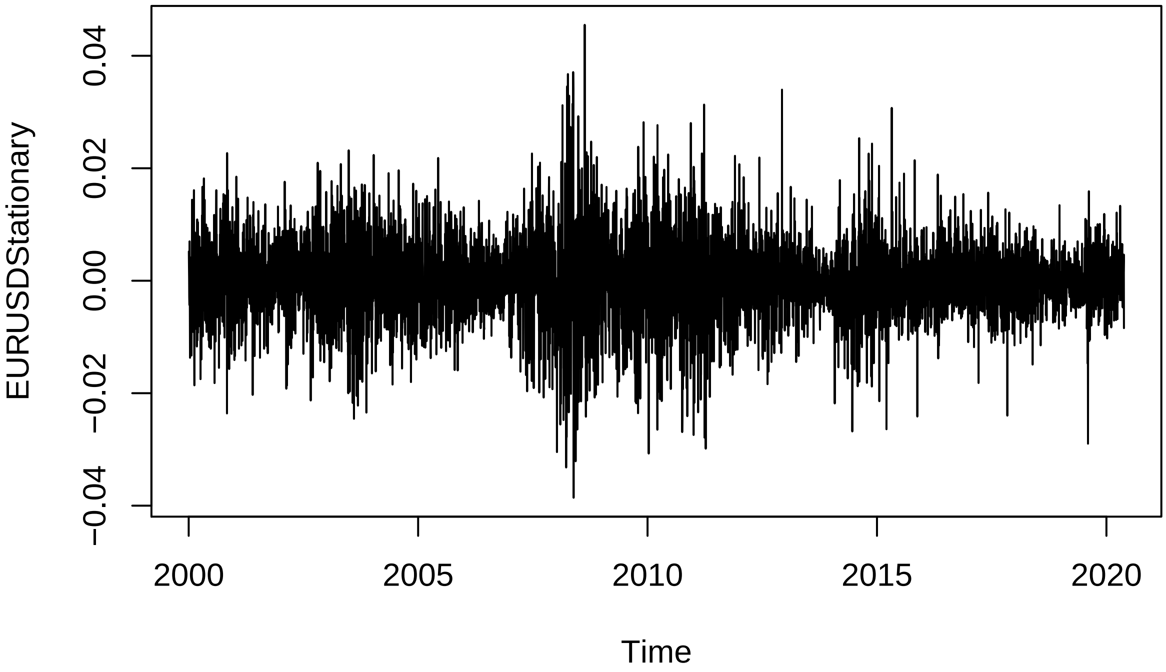
```
## Warning: package 'fUnitRoots' was built under R version 4.0.5

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



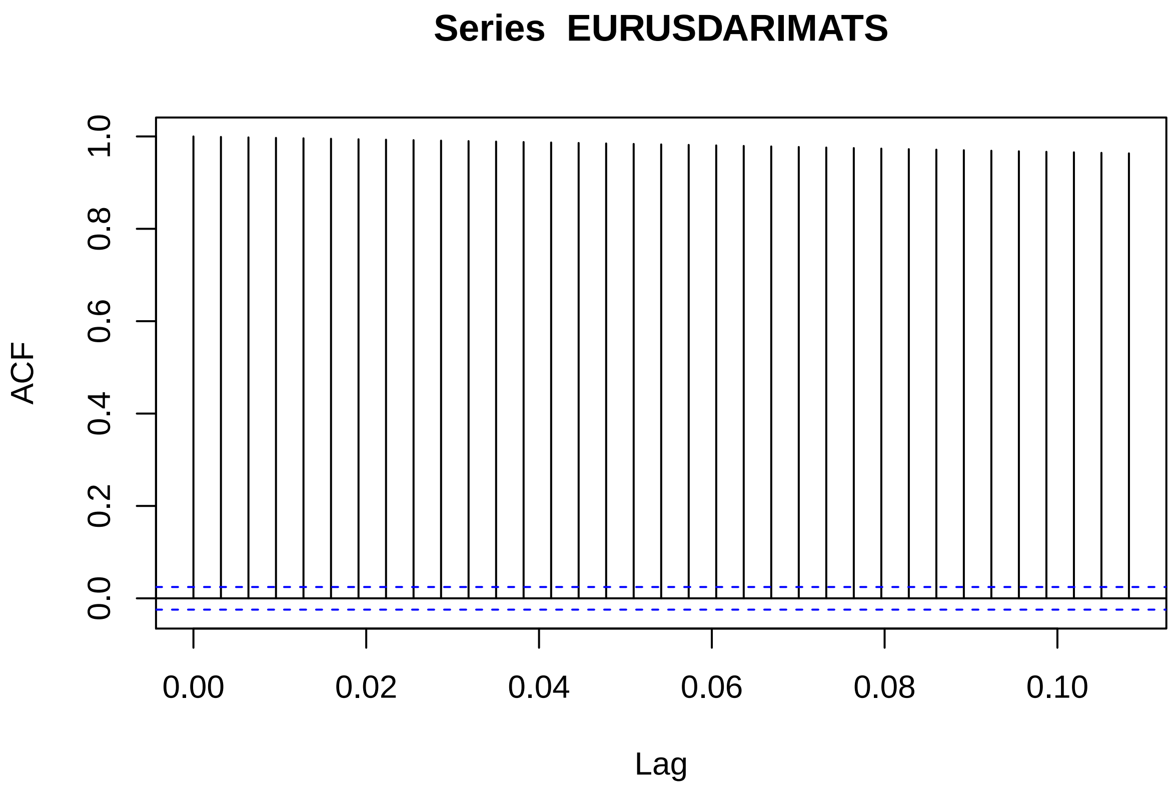
```
##
## Title:
##  KPSS Unit Root Test
##
## Test Results:
##   NA
##
## Description:
##  Tue May 04 00:11:16 2021 by user: janeo
```

```r
EURUSDStationary= diff(EURUSDARIMATS, differences=1)
plot(EURUSDStationary)
```

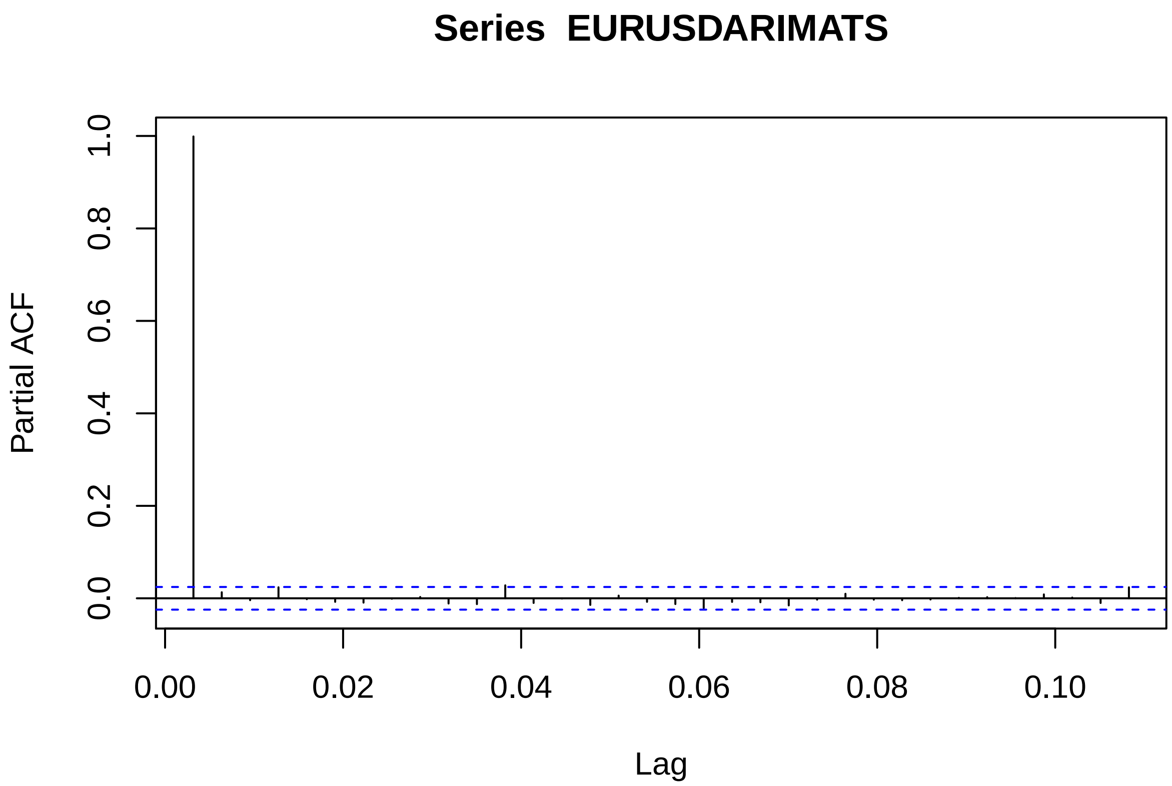


Calculating Autocorrlation function and partil autocorlation function

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

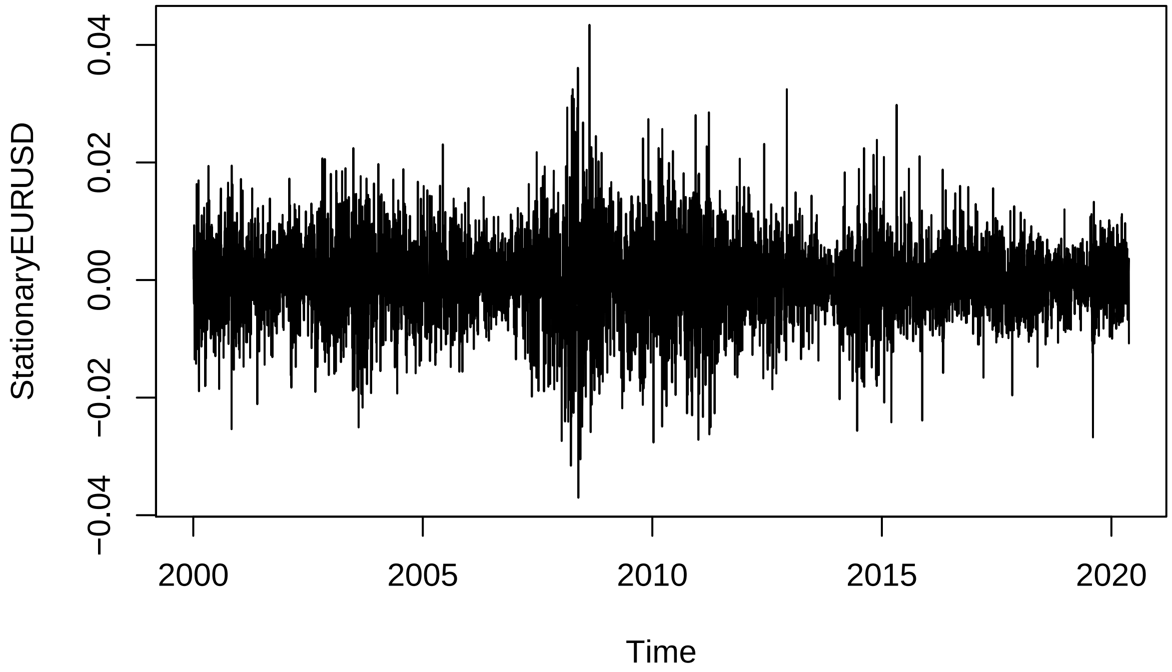


```
pacf(EURUSDARIMATS, lag.max = 34)
```



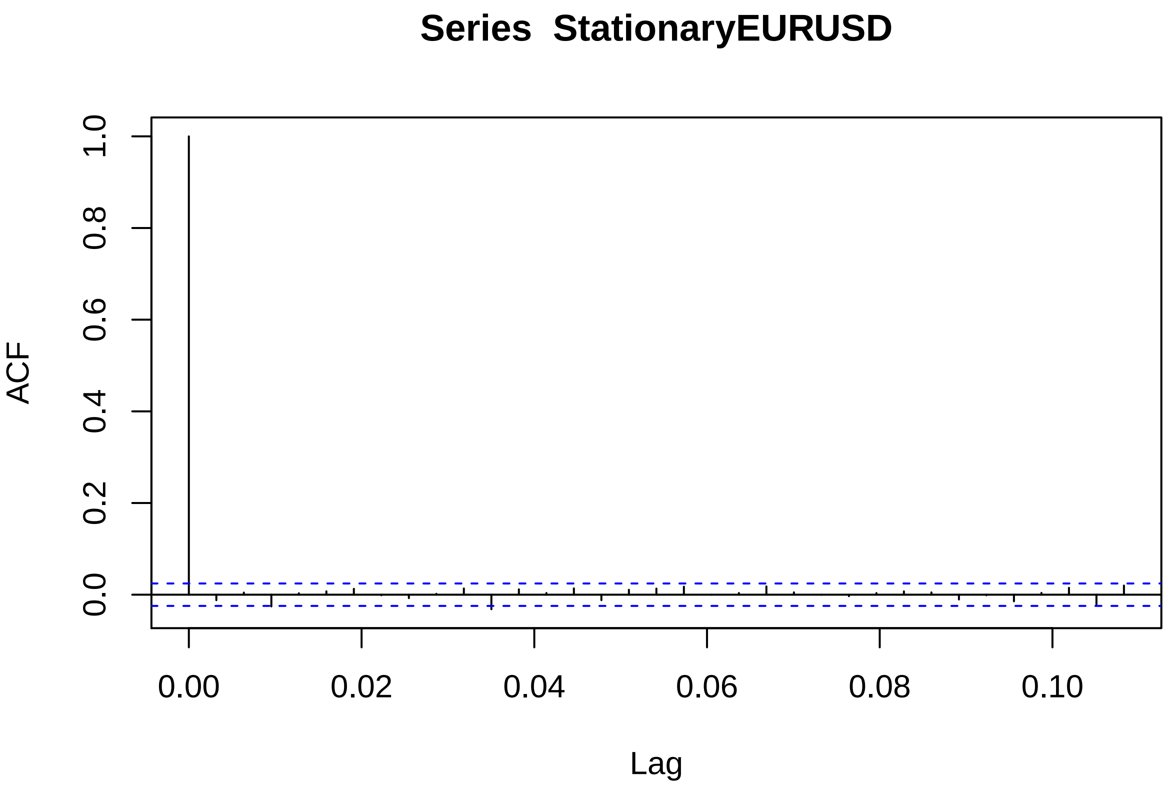
Adjusting and ensuring there are no seasonality

```
TSseasonallyadjustedEURUSD <- EURUSDARIMATS- ComponentEURUSD$seasonal
StationaryEURUSD <- diff(TSseasonallyadjustedEURUSD, differences=1)
plot(StationaryEURUSD)
```

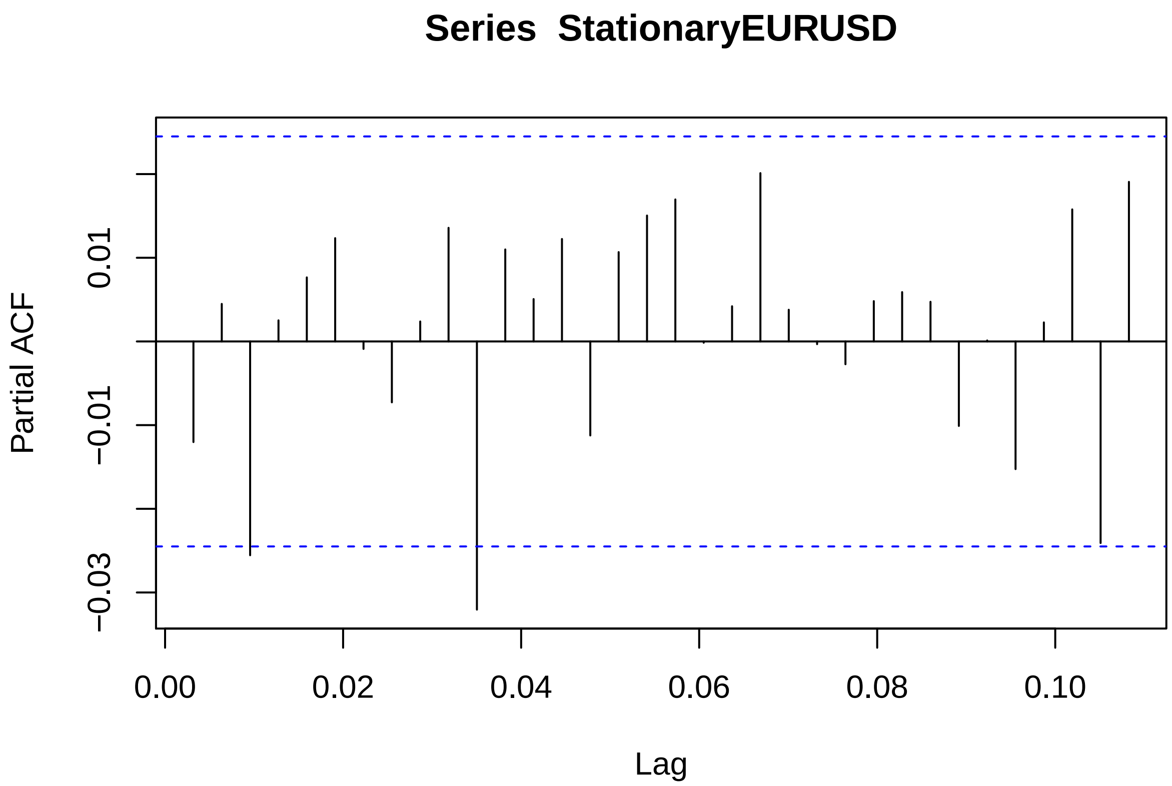


Calculating again for ACF and PACF after finding stationality

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



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



Fitting The ARIMA Model

ARIMA fitting (1,1,0)

```
fitArima1EURUSD <- arima(EURUSDARIMATS, order =  c(1,0,0), include.mean = TRUE)
fitArima1EURUSD
```

```
##
## Call:
## arima(x = EURUSDARIMATS, order = c(1, 0, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1  intercept
##       0.9990     1.2133
## s.e.  0.0005     0.0730
##
## sigma^2 estimated as 4.726e-05:  log likelihood = 22790.61,  aic = -45575.22
```

##Arima Fitting (0,1,0)

```
fitArima2EURUSD <- arima(EURUSDARIMATS, order =  c(0,1,0), include.mean = TRUE)
fitArima2EURUSD
```

```
##
## Call:
## arima(x = EURUSDARIMATS, order = c(0, 1, 0), include.mean = TRUE)
##
##
## sigma^2 estimated as 4.729e-05:  log likelihood = 22788.56,  aic = -45575.12
```

Arima Fitting (2,1,1)

```
fitArima3EURUSD <- arima(EURUSDARIMATS, order = c(2,1,1), include.mean = TRUE)
fitArima3EURUSD
```

```
##
## Call:
## arima(x = EURUSDARIMATS, order = c(2, 1, 1), include.mean = TRUE)
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##           ar1     ar2      ma1
##       -0.0062  0.0057  -0.0058
## s.e.      NaN  0.0103      NaN
##
## sigma^2 estimated as 4.728e-05:  log likelihood = 22789.14,  aic = -45570.28
```

##Fitting Arima (0,1,3)

```
fitArima4EURUSD <- arima(EURUSDARIMATS, order = c(3,1,0), include.mean = TRUE)
fitArima4EURUSD
```

```
##
## Call:
## arima(x = EURUSDARIMATS, order = c(3, 1, 0), include.mean = TRUE)
##
## Coefficients:
##           ar1     ar2      ar3
##       -0.0120  0.0055  -0.0270
## s.e.   0.0125  0.0125   0.0125
##
## sigma^2 estimated as 4.724e-05:  log likelihood = 22791.47,  aic = -45574.94
```

##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
```

```
ARIMAModelSelectionEURUSD = AIC(fitArima1EURUSD,fitArima2EURUSD,fitArima3EURUSD,fitArima4EURUSD)
```

```
## Warning in AIC.default(fitArima1EURUSD, fitArima2EURUSD, fitArima3EURUSD, :
## models are not all fitted to the same number of observations
```

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

```
##                 df       AIC
## fitArima1EURUSD  3 -45575.22
## fitArima2EURUSD  1 -45575.12
## fitArima4EURUSD  4 -45574.94
## fitArima3EURUSD  4 -45570.28
```

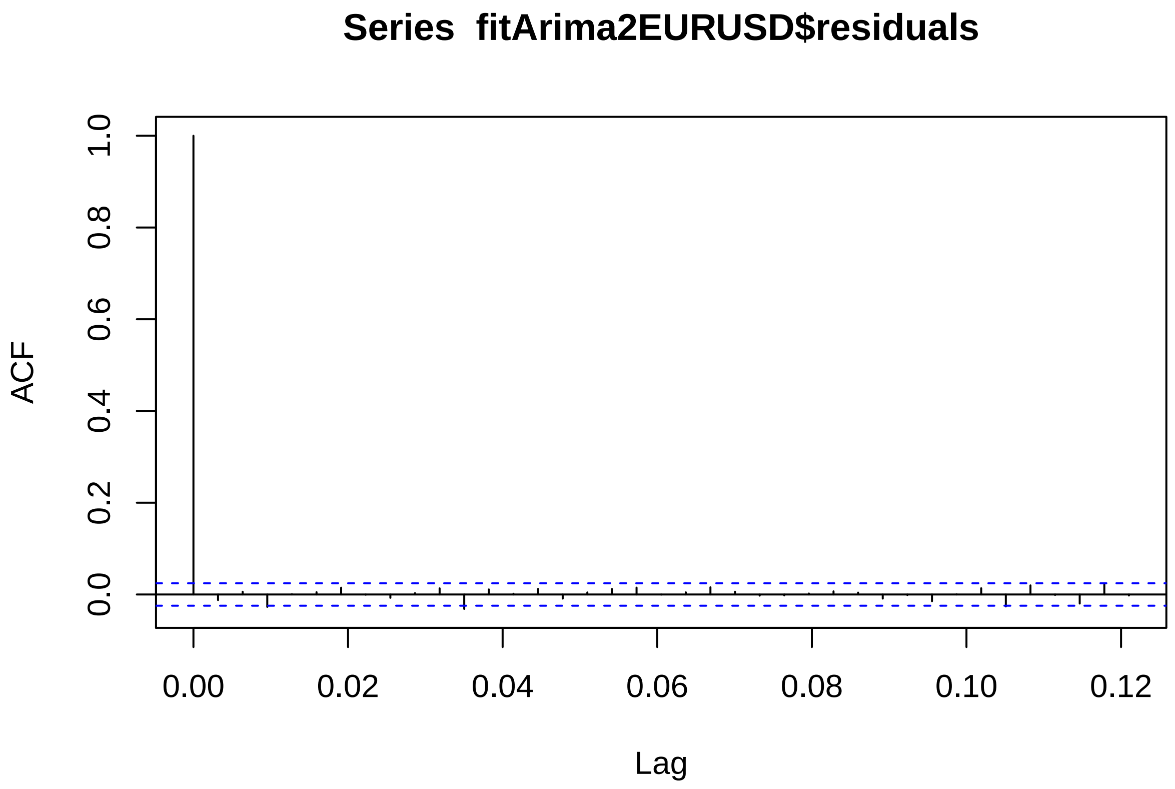
Base on the above the fitArima1CanJap is selected

```
confint(fitArima2EURUSD)
```

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

Runing code to obtain Box Test Rest

```
acf(fitArima2EURUSD$residuals)
```



```
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
```

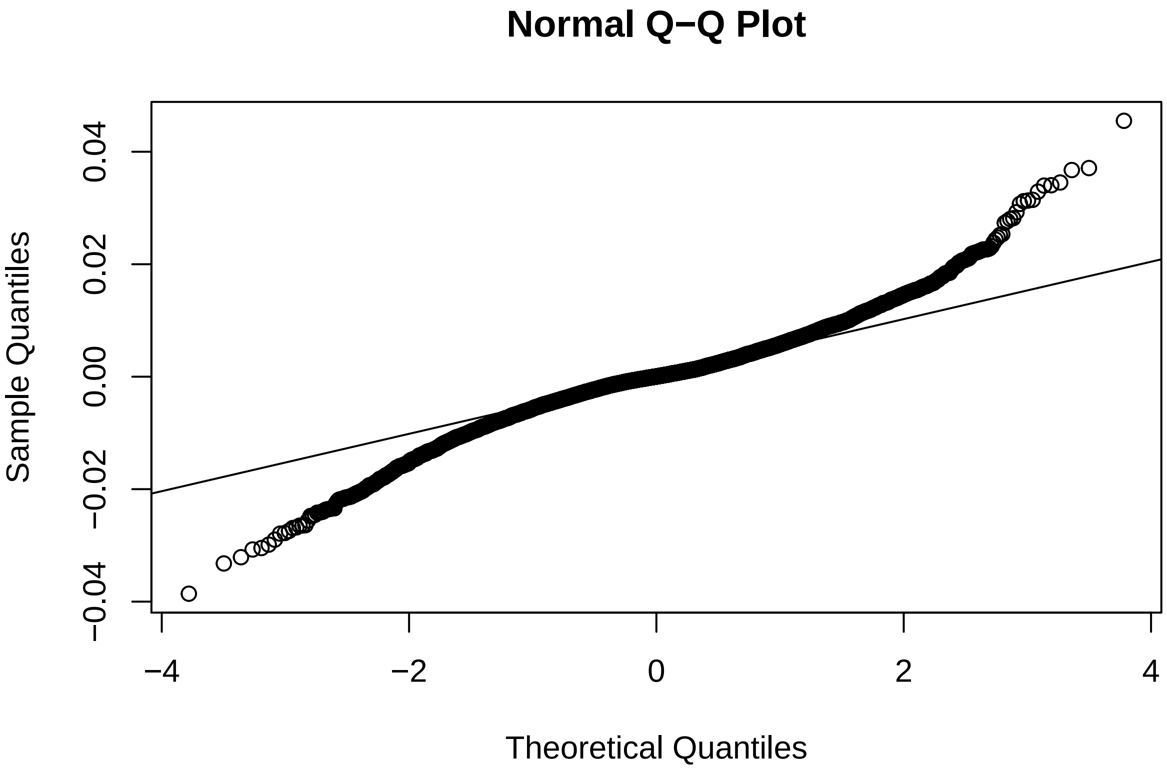
```
## Loading required package: bestglm
```

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

```
library(bestglm)
 Box.test(resid(fitArima2EURUSD),type="Ljung",lag=20,fitdf=1)
```

```
##
##  Box-Ljung test
##
## data:  resid(fitArima2EURUSD)
## X-squared = 20.284, df = 19, p-value = 0.3777
```

```
qqnorm(fitArima2EURUSD$residuals)
qqline(fitArima2EURUSD$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
```

```
auto.arima(EURUSDARIMATS, trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
## Error in polyroot(c(1, testvec)) : root finding code failed
##
##  ARIMA(2,1,2)(1,0,1)[314] with drift         : Inf
##  ARIMA(0,1,0)             with drift         : -45563.29
##  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)                                : -45565.16
##  ARIMA(0,1,0)(1,0,0)[314] with drift         : Inf
##  ARIMA(0,1,0)(0,0,1)[314] with drift         : -45561.35
##  ARIMA(0,1,0)(1,0,1)[314] with drift         : Inf
##  ARIMA(1,1,0)             with drift         : -45561.81
##  ARIMA(0,1,1)             with drift         : -45562.23
##  ARIMA(1,1,1)             with drift         : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,1,0)                                : -45575.12
##
##  Best model: ARIMA(0,1,0)
```

```
## Series: EURUSDARIMATS
## ARIMA(0,1,0)
##
## sigma^2 estimated as 4.729e-05:  log likelihood=22788.56
## AIC=-45575.12   AICc=-45575.12   BIC=-45568.36
```

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

```
forecastarimaEURUSD<- predict(fitArima2EURUSD,n.ahead = 100)
forecastarimaEURUSD
```

```
## $pred
## Time Series:
## Start = c(2020, 122)
## End = c(2020, 221)
## Frequency = 314
##   [1] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [10] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [19] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [28] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [37] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [46] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [55] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [64] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [73] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [82] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
##  [91] 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141 1.22141
## [100] 1.22141
##
## $se
## Time Series:
## Start = c(2020, 122)
## End = c(2020, 221)
## Frequency = 314
##   [1] 0.006876461 0.009724785 0.011910381 0.013752923 0.015376235 0.016843822
##   [7] 0.018193407 0.019449570 0.020629384 0.021745281 0.022806643 0.023820761
##  [13] 0.024793434 0.025729363 0.026632421 0.027505846 0.028352377 0.029174355
##  [19] 0.029973801 0.030752471 0.031511905 0.032253463 0.032978351 0.033687644
##  [25] 0.034382307 0.035063211 0.035731142 0.036386814 0.037030878 0.037663931
##  [31] 0.038286517 0.038899140 0.039502264 0.040096316 0.040681695 0.041258769
##  [37] 0.041827882 0.042389355 0.042943488 0.043490561 0.044030837 0.044564564
##  [43] 0.045091973 0.045613285 0.046128706 0.046638431 0.047142645 0.047641523
##  [49] 0.048135230 0.048623925 0.049107757 0.049586869 0.050061395 0.050531466
##  [55] 0.050997203 0.051458726 0.051916146 0.052369570 0.052819103 0.053264842
##  [61] 0.053706881 0.054145312 0.054580221 0.055011692 0.055439805 0.055864637
##  [67] 0.056286263 0.056704754 0.057120179 0.057532604 0.057942094 0.058348710
##  [73] 0.058752513 0.059153558 0.059551903 0.059947601 0.060340705 0.060731263
##  [79] 0.061119326 0.061504941 0.061888153 0.062269007 0.062647546 0.063023810
##  [85] 0.063397842 0.063769680 0.064139363 0.064506927 0.064872408 0.065235842
##  [91] 0.065597262 0.065956701 0.066314193 0.066669768 0.067023456 0.067375287
##  [97] 0.067725291 0.068073496 0.068419928 0.068764615
```

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