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

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

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

library(dplyr)  
library(lubridate)

##   
## Attaching package: 'lubridate'

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

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

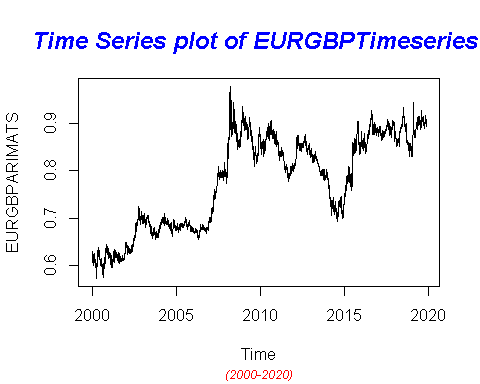
##Checking for obvious errors or missingg value

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

## integer(0)

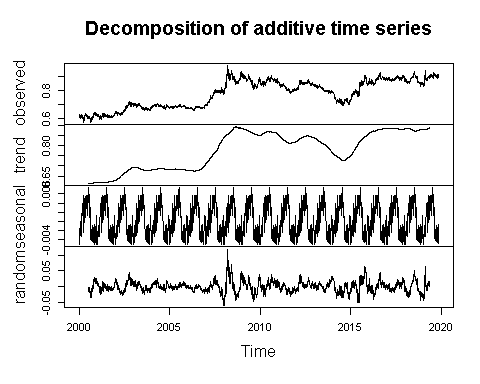
##Converting the data set into time series object

#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

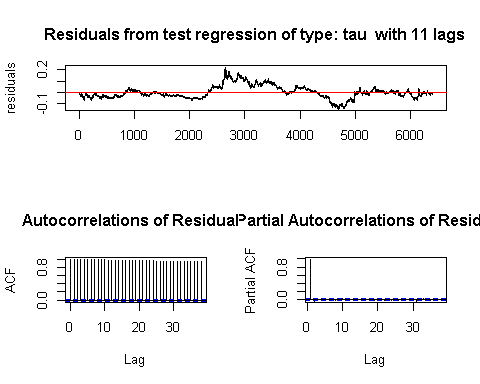
## Loading required package: timeSeries

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

## Loading required package: fBasics

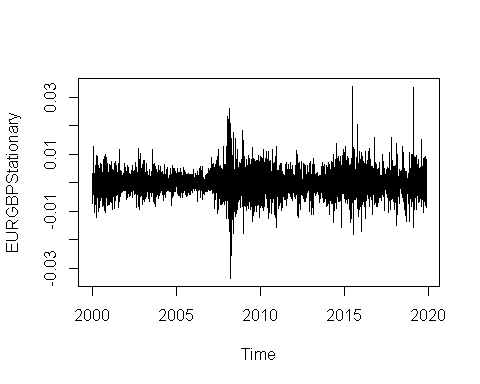
## Warning: package 'fBasics' was built under R version 4.0.5

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



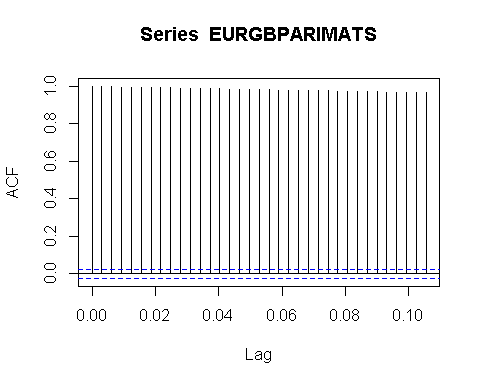
##   
## Title:  
## KPSS Unit Root Test  
##   
## Test Results:  
## NA  
##   
## Description:  
## Thu May 06 09:41:18 2021 by user: janeo

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

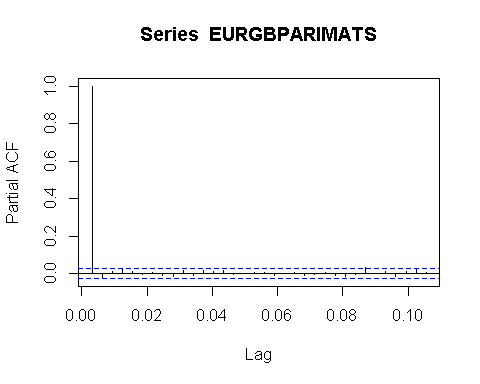


## Calculating Autocorrlation function and partil autocorlation function

acf(EURGBPARIMATS,lag.max=34)

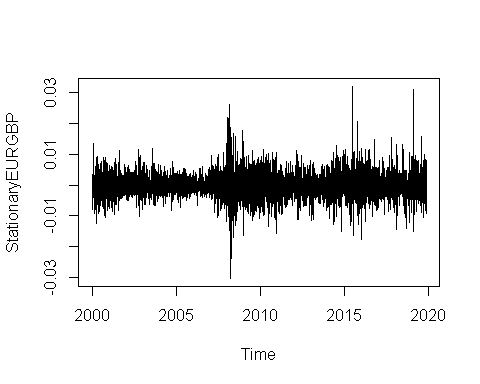


pacf(EURGBPARIMATS, lag.max = 34)



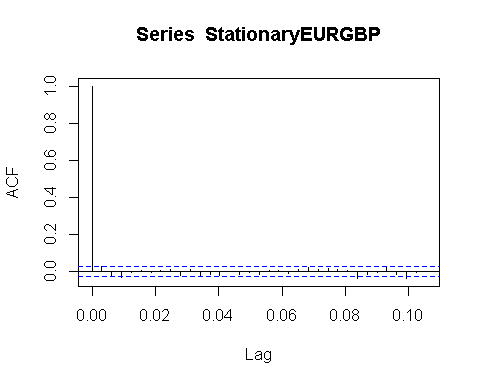
## Adjusting and ensuring there are no seasonality

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

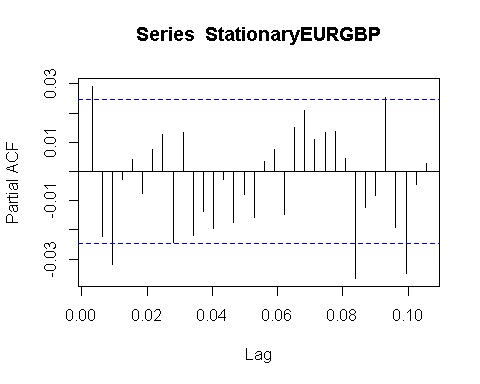


## Calculating again for ACF and PACF after finding stationality

acf(StationaryEURGBP, lag.max=34)



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 (4,0,4)

fitArima4EURGBP <- arima(EURGBPARIMATS, order = c(4,0,4), include.mean = TRUE)  
fitArima4EURGBP

##   
## Call:  
## arima(x = EURGBPARIMATS, order = c(4, 0, 4), include.mean = TRUE)  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ma1 ma2 ma3 ma4  
## 0.5779 0.1624 0.8452 -0.5861 0.4523 0.2507 -0.6186 -0.0413  
## s.e. 0.6550 0.0971 0.0800 0.5617 0.6539 0.6298 0.5637 0.0177  
## intercept  
## 0.7773  
## s.e. 0.0695  
##   
## sigma^2 estimated as 1.481e-05: log likelihood = 26501.12, aic = -52982.24

##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  
## fitArima3EURGBP 4 -52983.12  
## fitArima4EURGBP 10 -52982.24  
## fitArima2EURGBP 1 -52974.84  
## fitArima1EURGBP 3 -52974.11

### Base on the above the fitArima1CanJap is selected

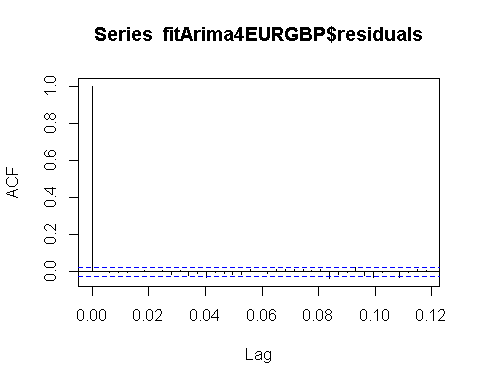
## 

confint(fitArima4EURGBP)

## 2.5 % 97.5 %  
## ar1 -0.70580122 1.861689401  
## ar2 -0.02800964 0.352710329  
## ar3 0.68841510 1.002052873  
## ar4 -1.68707498 0.514872558  
## ma1 -0.82942967 1.733933430  
## ma2 -0.98372493 1.485072692  
## ma3 -1.72353193 0.486277825  
## ma4 -0.07602829 -0.006645608  
## intercept 0.64103570 0.913647149

## Runing code to obtain Box Test Rest

acf(fitArima4EURGBP$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

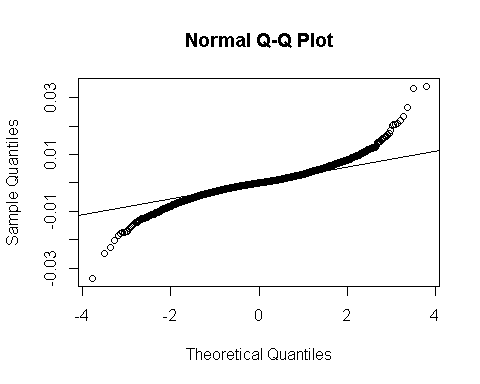
## Loading required package: bestglm

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

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

##   
## Box-Ljung test  
##   
## data: resid(fitArima4EURGBP)  
## X-squared = 16.997, df = 19, p-value = 0.5901

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

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(4,0,4)

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

## $pred  
## Time Series:  
## Start = c(2019, 283)   
## End = c(2020, 60)   
## Frequency = 322   
## [1] 0.8933922 0.8936537 0.8937878 0.8937870 0.8938391 0.8938292 0.8937527  
## [8] 0.8937514 0.8936993 0.8936101 0.8935938 0.8935266 0.8934403 0.8934181  
## [15] 0.8933440 0.8932639 0.8932374 0.8931595 0.8930860 0.8930554 0.8929754  
## [22] 0.8929077 0.8928728 0.8927920 0.8927293 0.8926901 0.8926094 0.8925508  
## [29] 0.8925075 0.8924277 0.8923723 0.8923250 0.8922467 0.8921936 0.8921428  
## [36] 0.8920663 0.8920149 0.8919609 0.8918865 0.8918361 0.8917794 0.8917072  
## [43] 0.8916573 0.8915983 0.8915284 0.8914785 0.8914177 0.8913499 0.8912997  
## [50] 0.8912375 0.8911717 0.8911210 0.8910578 0.8909939 0.8909423 0.8908785  
## [57] 0.8908163 0.8907638 0.8906997 0.8906389 0.8905855 0.8905213 0.8904618  
## [64] 0.8904074 0.8903434 0.8902848 0.8902295 0.8901658 0.8901081 0.8900519  
## [71] 0.8899886 0.8899315 0.8898746 0.8898118 0.8897551 0.8896975 0.8896354  
## [78] 0.8895789 0.8895208 0.8894592 0.8894029 0.8893444 0.8892834 0.8892271  
## [85] 0.8891682 0.8891079 0.8890516 0.8889924 0.8889326 0.8888763 0.8888169  
## [92] 0.8887576 0.8887012 0.8886418 0.8885829 0.8885263 0.8884669 0.8884085  
## [99] 0.8883517 0.8882924  
##   
## $se  
## Time Series:  
## Start = c(2019, 283)   
## End = c(2020, 60)   
## Frequency = 322   
## [1] 0.003847739 0.005524292 0.006750797 0.007726293 0.008583049 0.009357256  
## [7] 0.010048674 0.010700996 0.011316137 0.011885936 0.012437171 0.012964759  
## [13] 0.013462425 0.013949548 0.014418876 0.014866778 0.015307967 0.015734585  
## [19] 0.016145235 0.016551179 0.016944658 0.017325980 0.017703675 0.018070462  
## [25] 0.018427873 0.018782225 0.019126924 0.019464327 0.019798948 0.020124984  
## [31] 0.020445301 0.020762948 0.021072953 0.021378438 0.021681277 0.021977321  
## [37] 0.022269754 0.022559528 0.022843273 0.023124093 0.023402229 0.023675029  
## [43] 0.023945418 0.024213103 0.024476082 0.024737024 0.024995261 0.025249357  
## [49] 0.025501686 0.025751331 0.025997338 0.026241768 0.026483556 0.026722153  
## [55] 0.026959301 0.027193872 0.027425643 0.027656048 0.027883960 0.028109413  
## [61] 0.028333552 0.028555296 0.028774874 0.028993168 0.029209178 0.029423273  
## [67] 0.029636099 0.029846761 0.030055718 0.030263416 0.030469074 0.030673203  
## [73] 0.030876078 0.031077039 0.031276617 0.031474947 0.031671488 0.031866766  
## [79] 0.032060804 0.032253174 0.032444378 0.032634355 0.032822780 0.033010118  
## [85] 0.033196245 0.033380930 0.033564591 0.033747062 0.033928193 0.034108351  
## [91] 0.034287345 0.034465094 0.034641910 0.034817591 0.034992113 0.035165735  
## [97] 0.035338256 0.035509696 0.035680263 0.035849764

par(mfrow = c(1,1))