ARIMA EUR And USD

Jane

28/04/2021

# Forcasting Exchange Rate Using ARIMA Model for EUR And USD

## Reading EUR 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

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

library(dplyr)  
library(lubridate)

##   
## Attaching package: 'lubridate'

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

EURUSDARIMA$Date <- lubridate::ymd(EURUSDARIMA$Date)  
tail(EURUSDARIMA)

## Date RateEURUSD  
## 6396 2020-12-25 1.21922  
## 6397 2020-12-27 1.22038  
## 6398 2020-12-28 1.22196  
## 6399 2020-12-29 1.22519  
## 6400 2020-12-30 1.22984  
## 6401 2020-12-31 1.22141

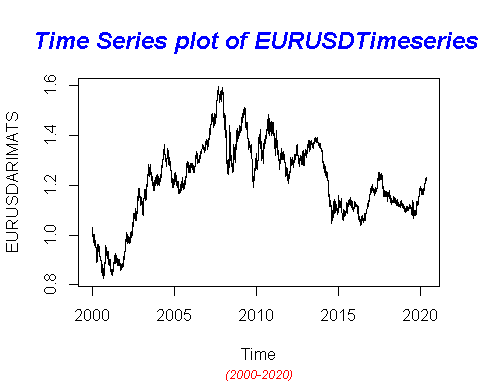
##Checking for obvious errors or missingg value

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

## integer(0)

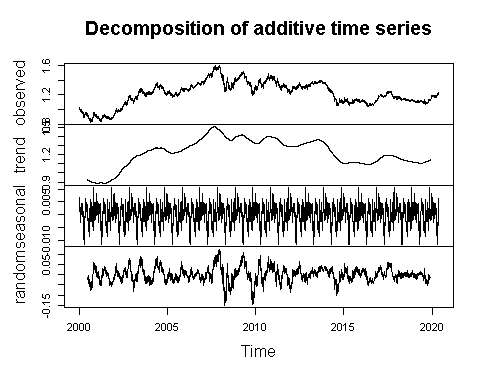
##Converting the data set into time series object

#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

ComponentEURUSDARIMA <- decompose(EURUSDARIMATS)  
plot(ComponentEURUSDARIMA)



## 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

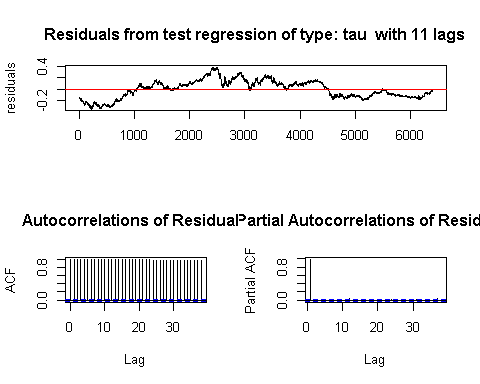
## Loading required package: timeSeries

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

## Loading required package: fBasics

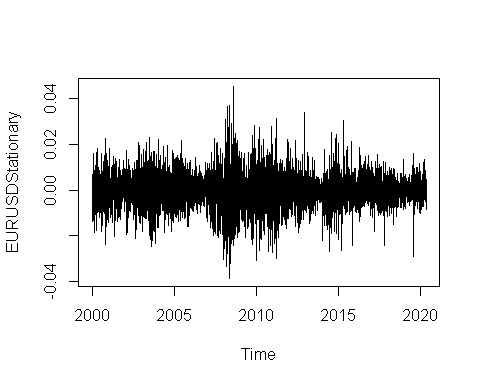
## Warning: package 'fBasics' was built under R version 4.0.5

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



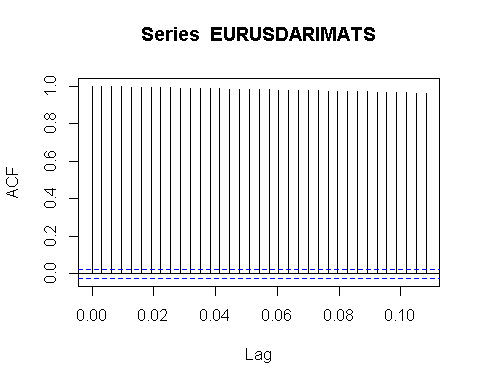
##   
## Title:  
## KPSS Unit Root Test  
##   
## Test Results:  
## NA  
##   
## Description:  
## Thu May 06 09:29:43 2021 by user: janeo

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

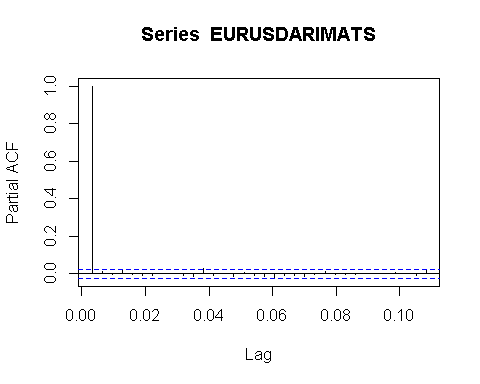


## Calculating Autocorrlation function and partil autocorlation function

acf(EURUSDARIMATS,lag.max=34)

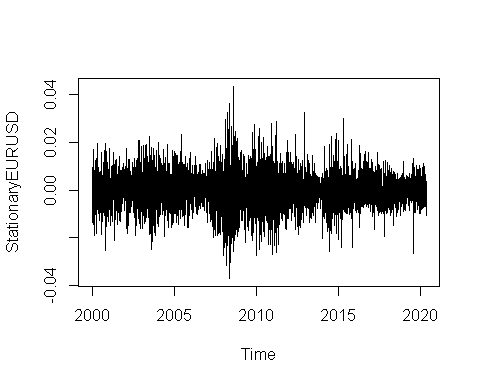


pacf(EURUSDARIMATS, lag.max = 34)



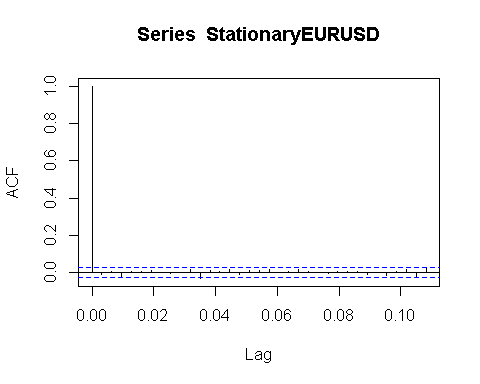
## Adjusting and ensuring there are no seasonality

TSseasonallyadjustedCanJap <- EURUSDARIMATS- ComponentEURUSDARIMA$seasonal   
StationaryEURUSD<- diff(TSseasonallyadjustedCanJap, differences=1)  
plot(StationaryEURUSD)

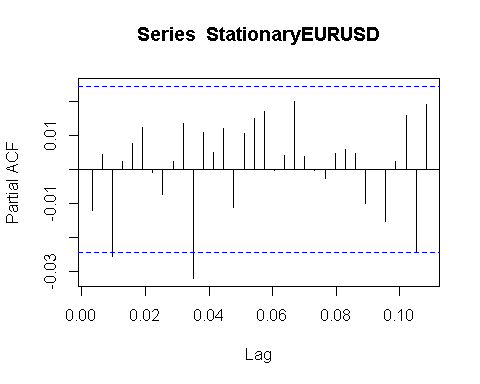


## Calculating again for ACF and PACF after finding stationality

acf(StationaryEURUSD, lag.max=34)



pacf(StationaryEURUSD, lag.max=34)



# Fitting The ARIMA Model

## ARIMA fitting (1,1,0)

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

##   
## Call:  
## arima(x = EURUSDARIMATS, order = c(1, 1, 0), include.mean = TRUE)  
##   
## Coefficients:  
## ar1  
## -0.0122  
## s.e. 0.0125  
##   
## sigma^2 estimated as 4.728e-05: log likelihood = 22789.04, aic = -45574.07

##Arima Fitting (1,1,1)

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

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

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

## ar1 ma1  
## -0.0061 -0.006  
## s.e. NaN NaN  
##   
## sigma^2 estimated as 4.728e-05: log likelihood = 22789.03, aic = -45572.07

## Arima Fitting (1,0,0)

fitArima3EURUSD <- arima(EURUSDARIMATS, order = c(1,0,0), include.mean = TRUE)  
fitArima3EURUSD

##   
## 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

##Fitting Arima (0,1,3)

fitArima4EURUSD <- arima(EURUSDARIMATS, order = c(2,2,2), include.mean = TRUE)  
fitArima4EURUSD

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

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

## ar1 ar2 ma1 ma2  
## -0.3148 0.0091 -0.697 -0.303  
## s.e. NaN 0.0104 NaN NaN  
##   
## sigma^2 estimated as 4.728e-05: log likelihood = 22781.22, aic = -45552.44

##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  
## fitArima3EURUSD 3 -45575.22  
## fitArima1EURUSD 2 -45574.07  
## fitArima2EURUSD 3 -45572.07  
## fitArima4EURUSD 5 -45552.44

### Base on the above the fitArima1CanJap is selected

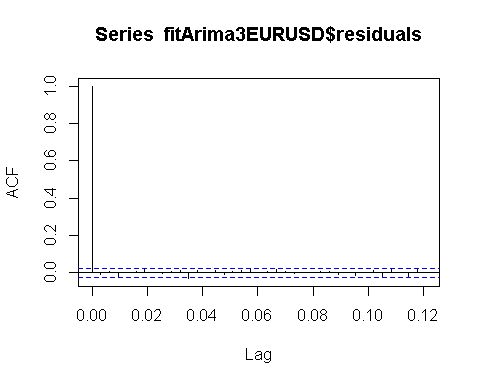
## 

confint(fitArima3EURUSD)

## 2.5 % 97.5 %  
## ar1 0.9979881 0.9999272  
## intercept 1.0702837 1.3564032

## Runing code to obtain Box Test Rest

acf(fitArima3EURUSD$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

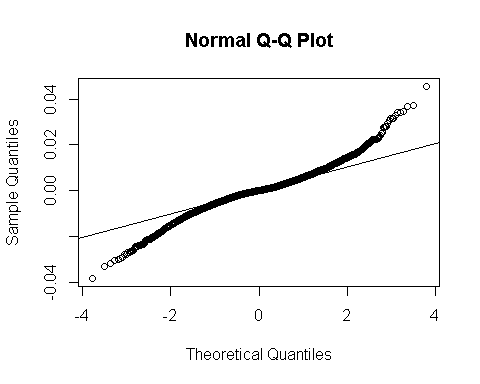
## Loading required package: bestglm

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

library(bestglm)  
 Box.test(resid(fitArima3EURUSD),type="Ljung",lag=20,fitdf=1)

##   
## Box-Ljung test  
##   
## data: resid(fitArima3EURUSD)  
## X-squared = 20.504, df = 19, p-value = 0.3649

qqnorm(fitArima3EURUSD$residuals)  
qqline(fitArima3EURUSD$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(1,1,0)

forecastarimaEURUSD<- predict(fitArima3EURUSD,n.ahead = 100)  
forecastarimaEURUSD

## $pred  
## Time Series:  
## Start = c(2020, 122)   
## End = c(2020, 221)   
## Frequency = 314   
## [1] 1.221402 1.221393 1.221385 1.221376 1.221368 1.221360 1.221351 1.221343  
## [9] 1.221335 1.221326 1.221318 1.221310 1.221301 1.221293 1.221285 1.221277  
## [17] 1.221268 1.221260 1.221252 1.221243 1.221235 1.221227 1.221219 1.221211  
## [25] 1.221202 1.221194 1.221186 1.221178 1.221170 1.221162 1.221153 1.221145  
## [33] 1.221137 1.221129 1.221121 1.221113 1.221105 1.221097 1.221088 1.221080  
## [41] 1.221072 1.221064 1.221056 1.221048 1.221040 1.221032 1.221024 1.221016  
## [49] 1.221008 1.221000 1.220992 1.220984 1.220976 1.220968 1.220960 1.220952  
## [57] 1.220944 1.220937 1.220929 1.220921 1.220913 1.220905 1.220897 1.220889  
## [65] 1.220881 1.220873 1.220866 1.220858 1.220850 1.220842 1.220834 1.220826  
## [73] 1.220819 1.220811 1.220803 1.220795 1.220788 1.220780 1.220772 1.220764  
## [81] 1.220757 1.220749 1.220741 1.220733 1.220726 1.220718 1.220710 1.220703  
## [89] 1.220695 1.220687 1.220680 1.220672 1.220664 1.220657 1.220649 1.220641  
## [97] 1.220634 1.220626 1.220619 1.220611  
##   
## $se  
## Time Series:  
## Start = c(2020, 122)   
## End = c(2020, 221)   
## Frequency = 314   
## [1] 0.006874766 0.009717322 0.011895042 0.013728059 0.015340448 0.016795875  
## [7] 0.018132182 0.019374039 0.020538591 0.021638326 0.022682680 0.023678986  
## [13] 0.024633077 0.025549688 0.026432722 0.027285443 0.028110613 0.028910595  
## [19] 0.029687427 0.030442887 0.031178530 0.031895730 0.032595709 0.033279556  
## [25] 0.033948249 0.034602668 0.035243612 0.035871807 0.036487912 0.037092534  
## [31] 0.037686227 0.038269502 0.038842832 0.039406653 0.039961370 0.040507358  
## [37] 0.041044970 0.041574532 0.042096350 0.042610711 0.043117883 0.043618121  
## [43] 0.044111662 0.044598729 0.045079537 0.045554283 0.046023159 0.046486344  
## [49] 0.046944008 0.047396313 0.047843414 0.048285455 0.048722578 0.049154916  
## [55] 0.049582594 0.050005735 0.050424455 0.050838865 0.051249071 0.051655174  
## [61] 0.052057274 0.052455463 0.052849832 0.053240467 0.053627452 0.054010866  
## [67] 0.054390787 0.054767289 0.055140444 0.055510321 0.055876987 0.056240506  
## [73] 0.056600939 0.056958349 0.057312791 0.057664323 0.058013000 0.058358873  
## [79] 0.058701994 0.059042412 0.059380176 0.059715331 0.060047923 0.060377996  
## [85] 0.060705592 0.061030753 0.061353518 0.061673927 0.061992017 0.062307825  
## [91] 0.062621388 0.062932741 0.063241916 0.063548948 0.063853868 0.064156708  
## [97] 0.064457499 0.064756270 0.065053051 0.065347870

par(mfrow = c(1,1))