Holt Winter and ARIMA on INR vs USD

Aakash Khandelwal - IPG2012001

May 9, 2016

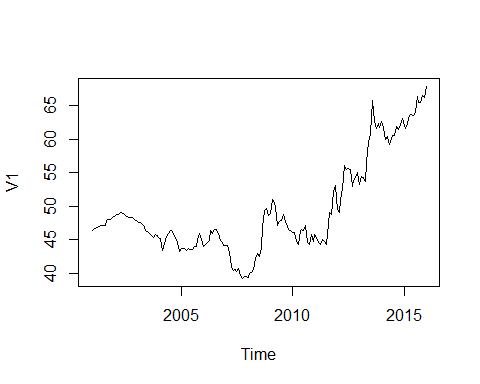
## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

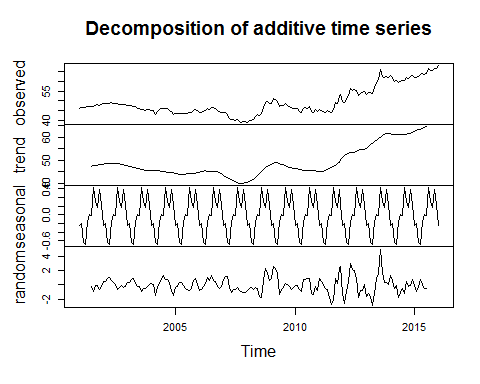
# Getting Data and plotting  
setwd("C:/Users/coffee/Downloads/Software/R/Cowpertwait/IF")  
data = read.csv("data.csv", header = FALSE)  
data.ts = ts(data, start = 2001, frequency = 12)  
head(data.ts)

## [1] 46.39 46.54 46.71 46.84 47.00 47.04

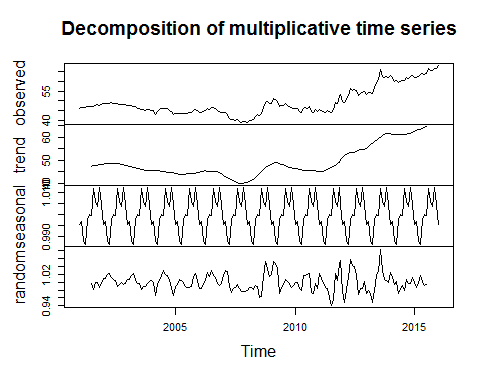
plot.ts(data.ts)



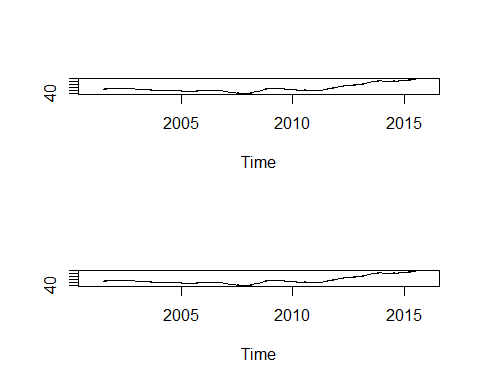
# time series decomposition (additive), into TREND, SEASON and RANDOM ERRORS  
a = decompose(data.ts, type="additive")  
plot(a)



# time series decomposition (multiplicative), into TREND, SEASON and RANDOM ERRORS  
b = decompose(data.ts, type="mult")  
plot(b)



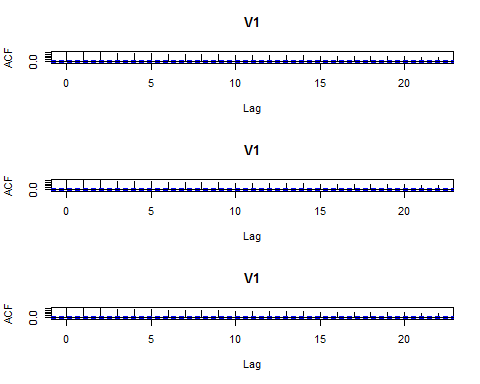
# Com#pare additive and multiplicative  
par(mfrow = c(2,1))  
ts.plot(cbind(a$trend, a$trend + a$seasonal), lty = 1:2)  
ts.plot(cbind(b$trend, b$trend \* b$seasonal), lty = 1:2)



# AUTOCORRELATION with lag 1 and 2 of orignal data  
par(mfrow = c(3,1))  
acf(data)  
acf(data)$acf[2]

## [1] 0.9677792

acf(data)$acf[3]



## [1] 0.936691

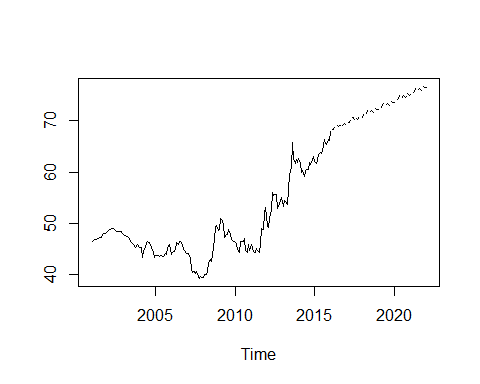
# HOLT WINTERS - including level, slope and seasonal effects - ADDITIVE  
data.hw4 = HoltWinters(data.ts, seasonal = "additive")  
data.hw4$coef

## a b s1 s2 s3 s4   
## 68.18042311 0.12055653 -0.01034713 -0.09673709 0.08736384 0.51218266   
## s5 s6 s7 s8 s9 s10   
## 0.39137121 -0.01278056 0.15154959 -0.38206687 -0.21855667 0.14821692   
## s11 s12   
## -0.24200736 -0.30242311

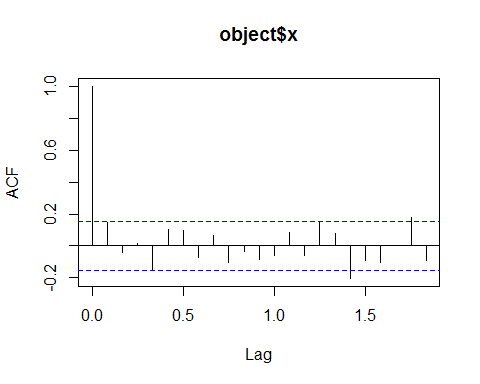
data.hw4$SSE

## [1] 250.0229

par(mfrow = c(1,1))  
data.predict1 = predict(data.hw4, n.ahead = 6\*12)  
ts.plot(data.ts, data.predict1, lty = 1:2)



acf(residuals(data.hw4))



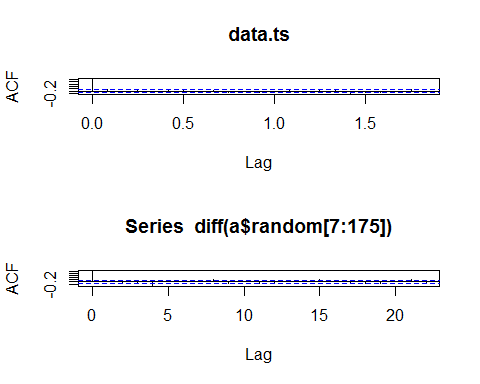
# Calculating if residual series (data.ts - data.hw4$fitted[,1]) is white noise or not  
residual\_series = data.ts - data.hw4$fitted[,1]  
mean(residual\_series)

## [1] 0.002901398

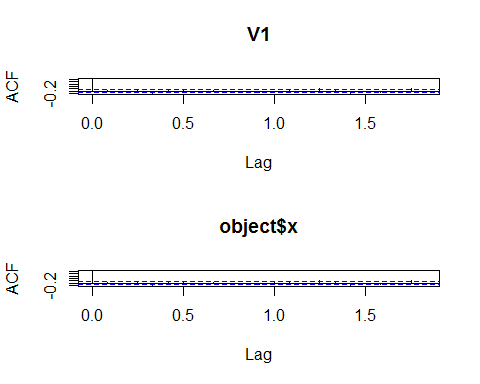
var(residual\_series)

## data.ts  
## data.ts 1.488223

par(mfrow=c(2,1))  
acf(residual\_series)  
# variance = 1.5, mean = 0, and correlogram shows it resembles white noise.  
acf(diff(a$random[7:175]))



# first order difference of random walk are white noise  
# shows that random errors follows random walk  
# head(a$random[7:175])  
# head(diff(a$random[7:175]))  
acf(diff(data.ts))  
# first order differences also shows no significant value at lag 1 but other significant values shows the model needs some extension.  
data.hw4\_2 = HoltWinters(data.ts, gamma=0)  
acf(resid(data.hw4\_2))



# better fit using gamma =0  
  
# Fitted AR  
layout(1:1)  
data.ar = ar(data.ts)  
mean(data.ts)

## [1] 49.4268

data.ar$order

## [1] 1

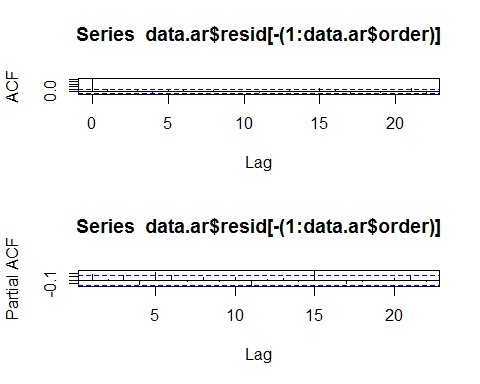
data.ar$ar

## [1] 0.9677792

data.ar$ar + c(-1.96,1.96)\*sqrt(data.ar$asy.var)

## [1] 0.9308912 1.0046673

par(mfrow=c(2,1))  
acf(data.ar$resid[-(1:data.ar$order)])  
pacf(data.ar$resid[-(1:data.ar$order)])



#method = MLE, only take vector i.e first column  
data.ar2 = ar(data.ts[,1], method = "mle")

## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1  
  
## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1  
  
## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1  
  
## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1  
  
## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1  
  
## Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible  
## convergence problem: optim gave code = 1

mean(data.ts)

## [1] 49.4268

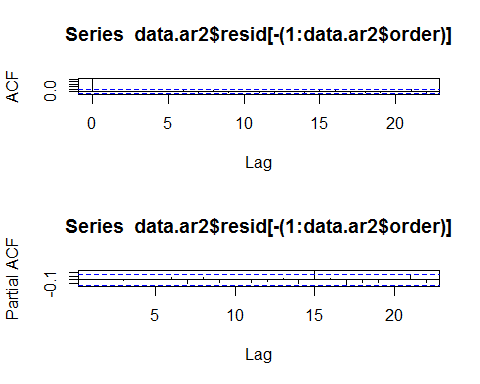
data.ar2$order

## [1] 7

data.ar2$ar

## [1] 1.12327492 -0.17770508 0.14730567 -0.29403681 0.35168902 -0.03855551  
## [7] -0.12122875

acf(data.ar2$resid[-(1:data.ar2$order)])  
pacf(data.ar2$resid[-(1:data.ar2$order)])



# fitting moving aveergae  
data.ma = arima(data.ts, order = c(0,0,1))  
data.ma

##   
## Call:  
## arima(x = data.ts, order = c(0, 0, 1))  
##   
## Coefficients:  
## ma1 intercept  
## 0.9426 49.4671  
## s.e. 0.0235 0.5548  
##   
## sigma^2 estimated as 14.84: log likelihood = -502.05, aic = 1010.09

par(mfrow=c(2,1))  
acf(data.ma$res[-1])  
# not a satisfactory fit  
# com#parison of ARMA, AR, and MA  
  
data.ARvsARMA = arima(data.ts, order = c(1,0,0))  
data.MAvsARMA = arima(data.ts, order = c(0,0,1))  
data.ARMA = arima(data.ts, order = c(1,0,1))  
AIC(data.ARvsARMA)

## [1] 569.9773

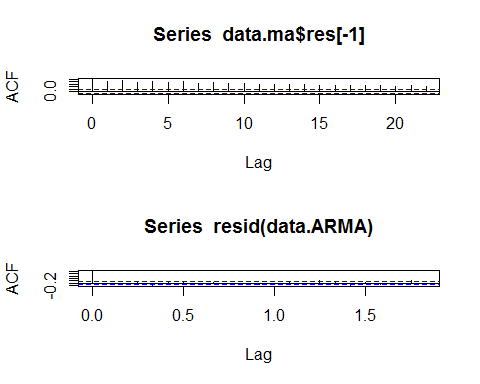
AIC(data.MAvsARMA)

## [1] 1010.094

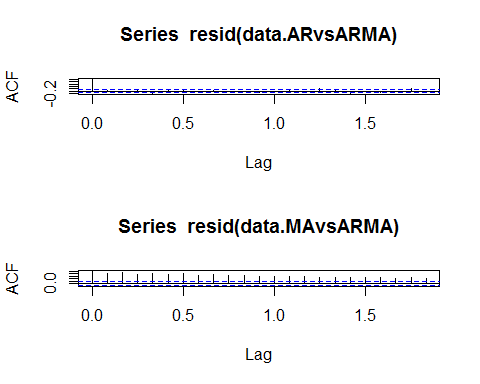
AIC(data.ARMA)

## [1] 570.2117

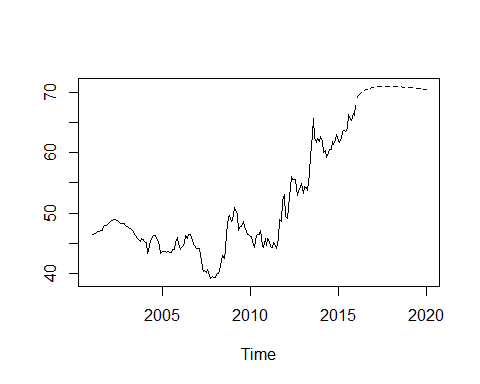
acf(resid(data.ARMA))



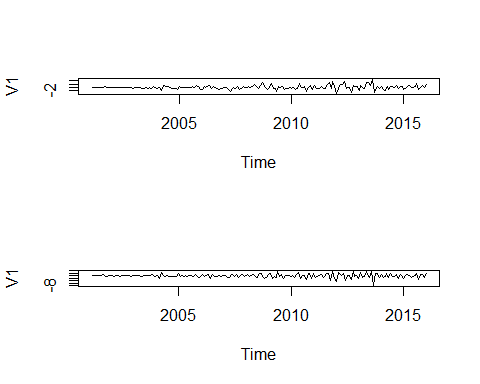
acf(resid(data.ARvsARMA))  
acf(resid(data.MAvsARMA))



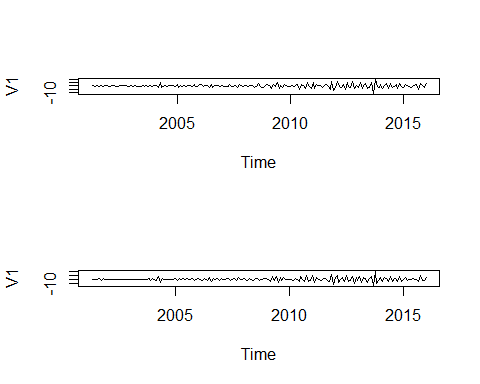
# both AR and ARMA are good approximate  
predict.arma = predict(data.ARMA, n.ahead = 48)  
par(mfrow=c(1,1))  
ts.plot(cbind(data.ts, predict.arma$pred+predict.arma$se), lty = 1:2)



# difference  
par(mfrow=c(2,1))  
  
plot(diff(data.ts))  
plot(diff(data.ts, d = 2))



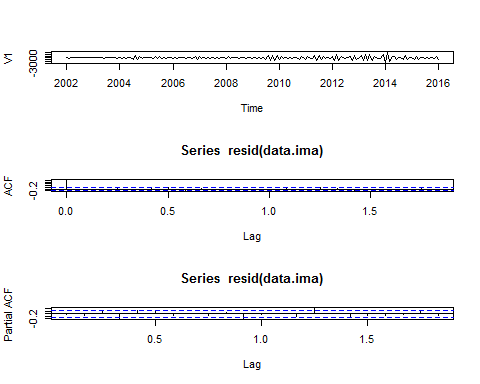
plot(diff(data.ts, d = 3))  
  
plot(diff(data.ts, d = 4))



par(mfrow=c(3,1))  
  
plot(diff(data.ts, d = 12))  
  
#ARIMA  
data.ima = arima(data.ts, order = c(0,1,1))  
data.ima

##   
## Call:  
## arima(x = data.ts, order = c(0, 1, 1))  
##   
## Coefficients:  
## ma1  
## 0.1012  
## s.e. 0.0798  
##   
## sigma^2 estimated as 1.281: log likelihood = -277.7, aic = 559.41

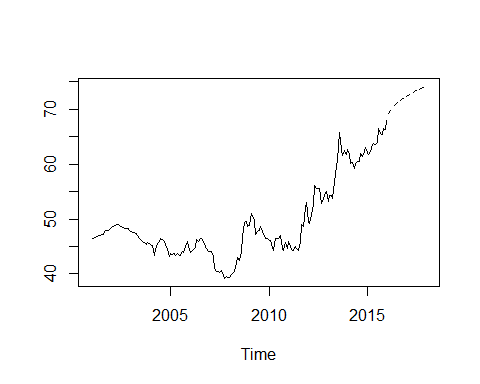
acf(resid(data.ima))  
pacf(resid(data.ima))



par(mfrow=c(1,1))  
data.ima\_predict = predict(data.ima, n.ahead = 24)  
data.ima\_predict

## $pred  
## Jan Feb Mar Apr May Jun Jul  
## 2016 68.05066 68.05066 68.05066 68.05066 68.05066 68.05066  
## 2017 68.05066 68.05066 68.05066 68.05066 68.05066 68.05066 68.05066  
## 2018 68.05066   
## Aug Sep Oct Nov Dec  
## 2016 68.05066 68.05066 68.05066 68.05066 68.05066  
## 2017 68.05066 68.05066 68.05066 68.05066 68.05066  
## 2018   
##   
## $se  
## Jan Feb Mar Apr May Jun Jul  
## 2016 1.131821 1.683567 2.094702 2.437449 2.737615 3.007975  
## 2017 4.285812 4.463358 4.634107 4.798784 4.957994 5.112248 5.261983  
## 2018 6.083481   
## Aug Sep Oct Nov Dec  
## 2016 3.255963 3.486355 3.702439 3.906588 4.100586  
## 2017 5.407573 5.549344 5.687583 5.822540 5.954440  
## 2018

ts.plot(cbind(data.ts, data.ima\_predict$pred+data.ima\_predict$se), lty = 1:2)



data.ima\_forecast = forecast(data.ima)  
data.ima\_forecast

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Feb 2016 68.05066 66.60017 69.50114 65.83233 70.26899  
## Mar 2016 68.05066 65.89308 70.20823 64.75093 71.35039  
## Apr 2016 68.05066 65.36619 70.73513 63.94512 72.15620  
## May 2016 68.05066 64.92694 71.17437 63.27334 72.82797  
## Jun 2016 68.05066 64.54226 71.55905 62.68503 73.41628  
## Jul 2016 68.05066 64.19578 71.90553 62.15513 73.94618  
## Aug 2016 68.05066 63.87797 72.22334 61.66909 74.43223  
## Sep 2016 68.05066 63.58271 72.51860 61.21753 74.88379  
## Oct 2016 68.05066 63.30579 72.79552 60.79401 75.30730  
## Nov 2016 68.05066 63.04416 73.05715 60.39388 75.70743  
## Dec 2016 68.05066 62.79554 73.30577 60.01365 76.08766  
## Jan 2017 68.05066 62.55817 73.54315 59.65062 76.45069  
## Feb 2017 68.05066 62.33063 73.77068 59.30263 76.79868  
## Mar 2017 68.05066 62.11181 73.98950 58.96797 77.13334  
## Apr 2017 68.05066 61.90077 74.20055 58.64521 77.45610  
## May 2017 68.05066 61.69673 74.40458 58.33317 77.76815  
## Jun 2017 68.05066 61.49905 74.60227 58.03083 78.07048  
## Jul 2017 68.05066 61.30715 74.79416 57.73736 78.36395  
## Aug 2017 68.05066 61.12057 74.98074 57.45201 78.64930  
## Sep 2017 68.05066 60.93889 75.16243 57.17414 78.92717  
## Oct 2017 68.05066 60.76173 75.33959 56.90320 79.19811  
## Nov 2017 68.05066 60.58877 75.51254 56.63869 79.46263  
## Dec 2017 68.05066 60.41973 75.68158 56.38017 79.72114  
## Jan 2018 68.05066 60.25436 75.84695 56.12725 79.97406

plot(data.ima\_forecast)

