1—Atajan-Abdyyev.R

Atajan Abdyyev

2022-06-12

#Author: Atajan Abdyyev  
  
#1 Check you working directory  
#getwd()

#2 Set your working directory to "ANLY 565/RScript"  
setwd("C:/Users/ataja/OneDrive/Desktop/ANLY 565/RScript")  
  
#3 Download goy data set posted on Canvas and lable it   
# goy. This dataset reperesnets daily prices of gold,  
# oil, and the price of 1 US dollar in terms of Japanese yen.  
# Set the first column in each data set to the date format   
# and the remaining columns in numerical format.  
library("readxl")  
goy = read\_excel("goy.xls", col\_names = TRUE, col\_types = c("date", "numeric", "numeric", "numeric"))  
  
  
  
goy$observation\_date = as.Date(goy$observation\_date)  
  
str(goy)

## tibble [879 x 4] (S3: tbl\_df/tbl/data.frame)  
## $ observation\_date: Date[1:879], format: "1946-01-01" "1946-02-01" ...  
## $ gold : num [1:879] NA NA NA NA NA NA NA NA NA NA ...  
## $ oil : num [1:879] 1.17 1.17 1.17 1.27 1.27 1.27 1.27 1.52 1.52 1.52 ...  
## $ yen : num [1:879] NA NA NA NA NA NA NA NA NA NA ...

#4 Create a new data set called "goycc" that contains all complete cases of goy data.  
# Utilize complete.cases function.  
goycc = goy[complete.cases(goy), ]  
  
str(goycc)

## tibble [578 x 4] (S3: tbl\_df/tbl/data.frame)  
## $ observation\_date: Date[1:578], format: "1971-01-01" "1971-02-01" ...  
## $ gold : num [1:578] 37.9 38.7 38.9 39 40.5 ...  
## $ oil : num [1:578] 3.56 3.56 3.56 3.56 3.56 3.56 3.56 3.56 3.56 3.56 ...  
## $ yen : num [1:578] 358 358 358 358 357 ...

#5 Create a stand alone variable "date" that takes on values of "observation\_date"  
# variable from the goycc data set. Set the mode of the variable to character  
date = as.character(goycc$observation\_date)  
  
#6 Find the range of dates covered in goycc data set by applying range function  
# to "date" variable.   
range(date)

## [1] "1971-01-01" "2019-02-01"

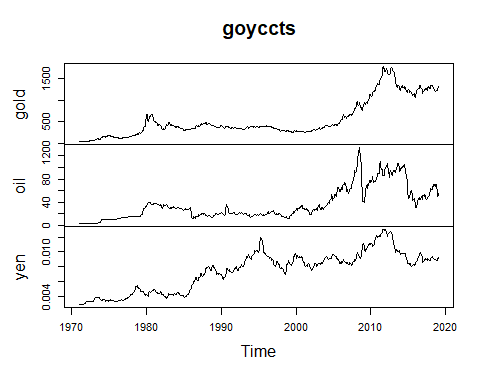
#7 Create a time series object called "goyccts" by utilizing goycc dataset and   
# ts() function. In this time series object please exclude the first column   
# of the goycc dataset.   
goyccts = ts(goycc[,2:4], start = c(1971, 1), freq = 12)  
str(goyccts)

## Time-Series [1:578, 1:3] from 1971 to 2019: 37.9 38.7 38.9 39 40.5 ...  
## - attr(\*, "dimnames")=List of 2  
## ..$ : NULL  
## ..$ : chr [1:3] "gold" "oil" "yen"

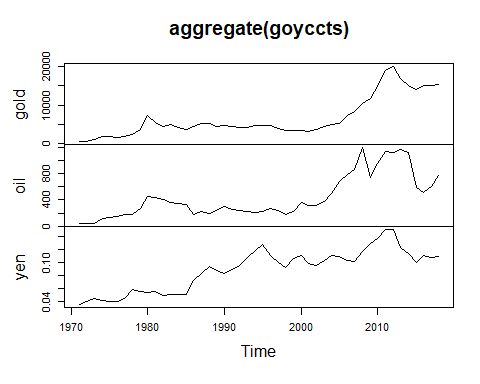
#8 Reassign the value of the yen variable from the goyccts data set  
# by converting the exchange rate of yen that represents   
# the price of 1 US Dollar in terms of Japanese yen to represent   
# the price of 1 Yen in terms of US Dollar.   
# This way if the number increases it represents appreciation of Yen.   
# Hint: Reassign the value of yen variable by taking a reciprocal   
goyccts[,'yen'] = 1/goyccts[,'yen']  
head(goyccts[,'yen'])

## [1] 0.002793140 0.002796851 0.002797057 0.002797178 0.002797884 0.002797893

#9 Plot the time series plot of the three assets. Do you see any trend?  
# Do you see any seasonal component?  
plot(goyccts)



#Answer: Trend shows upward across all variables and positive.  
  
  
#10 Utilize the aggregate function to plot annual average prices of the three assets.  
# How does this graph differ from the monthly time series plot?  
  
plot(aggregate(goyccts))



#Answer: Plot is smoother since it is aggregated and shows monthly average throught years,  
# make it easier to see trend on monthly average level.  
  
  
#11 Find the average summer price of oil for the entire sample.  
  
oil\_jun = window(goyccts[,'oil'], start = c(1971,6), freq = TRUE)  
oil\_jul = window(goyccts[,'oil'], start = c(1971,7), freq = TRUE)  
oil\_aug = window(goyccts[,'oil'], start = c(1971,8), freq = TRUE)  
  
oil\_summer = c(oil\_jun,oil\_jul,oil\_aug)  
mean(oil\_summer)

## [1] 37.26092

#Answer: oil average is 37.26 for summer time  
  
#12 Find the average winter price of oil for the entire sample.  
oil\_dec = window(goyccts[,'oil'], start = c(1971,12), freq = TRUE)  
oil\_jan = window(goyccts[,'oil'], start = c(1971,1), freq = TRUE)  
oil\_feb = window(goyccts[,'oil'], start = c(1971,2), freq = TRUE)  
  
oil\_winter = c(oil\_dec,oil\_jan,oil\_feb)  
mean(oil\_winter)

## [1] 34.74591

#Answer: Average for winter time is 34.75  
  
#13 Find how the summer price of oil compares to the winter price of oil.  
# Please provide your answer in percentages.   
summer\_comparison = (mean(oil\_summer)/mean(goyccts[,'oil']))  
paste('Summer price of oil comparison to enteire years:',summer\_comparison\*100,'%')

## [1] "Summer price of oil comparison to enteire years: 102.773294036483 %"

winter\_comparison = (mean(oil\_winter)/mean(goyccts[,'oil']))  
paste('Winter price of oil comparison to enteire years:',winter\_comparison\*100,'%')

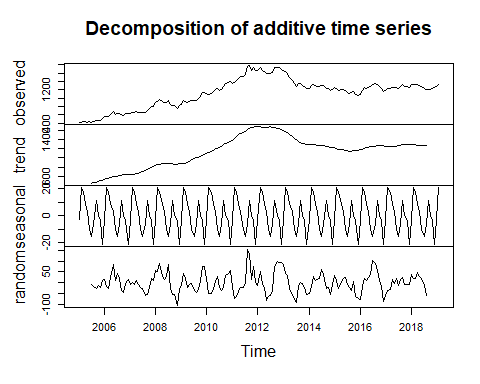
## [1] "Winter price of oil comparison to enteire years: 95.8363716588087 %"

#Answer: Summer price of oil comparison to enteire years: 102.77%  
#ANswer: Winter price of oil comparison to enteire years: 95.84%  
#Answer difference is that summer oil prices are ~7% higher than winter months (102-95)=7.  
  
#14 Use window() function to create three stand alone variables   
# "gold", "oil", and "yen" that take on values of the "gold", "oil", and "yen"   
# variables from the goyccts dataset starting from January of 2005  
gold = window(goyccts[,'gold'],start=c(2005,1))  
oil = window(goyccts[,'oil'],start=c(2005,1))  
yen = window(goyccts[,'yen'],start=c(2005,1))  
  
  
#15 Use plot and decompose functions to generate three graphs that would depict  
# the observed values, trends, seasonal, and random components for "gold"  
# "oil" and "yen" variables. Would you choose multiplicative or   
# additive decomposition model for each of the variables?  
layout(1:3)  
  
plot(gold)  
plot(oil)  
plot(yen)

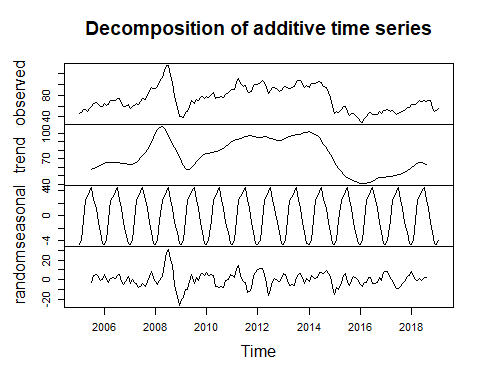
Chart

Description automatically generated

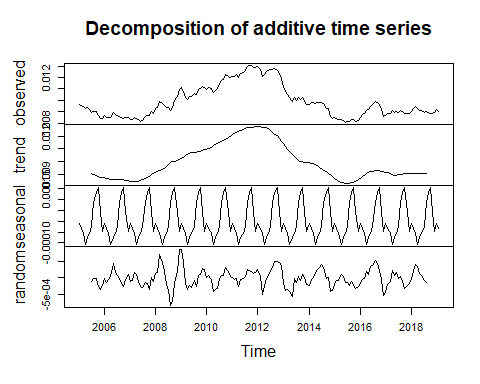
gold.decom.additive <- decompose(gold, type = "additive")  
oil.decom.additive <- decompose(oil, type = "additive")  
yen.decom.additive <- decompose(yen, type = "additive")  
  
plot(gold.decom.additive)



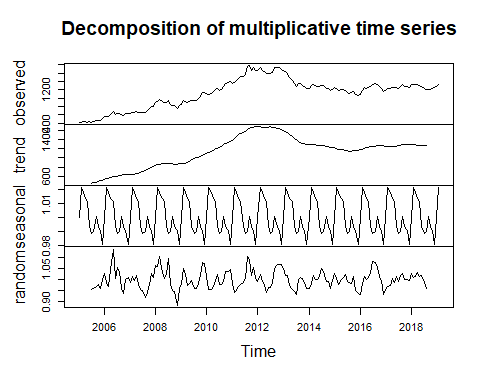
plot(oil.decom.additive)



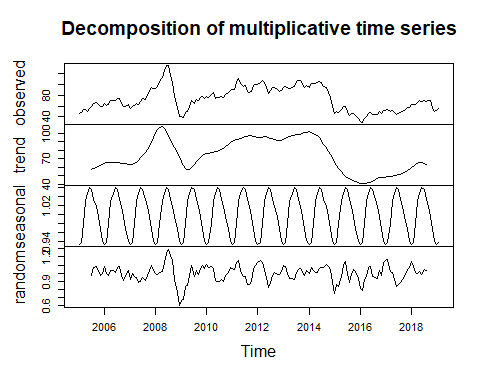
plot(yen.decom.additive)



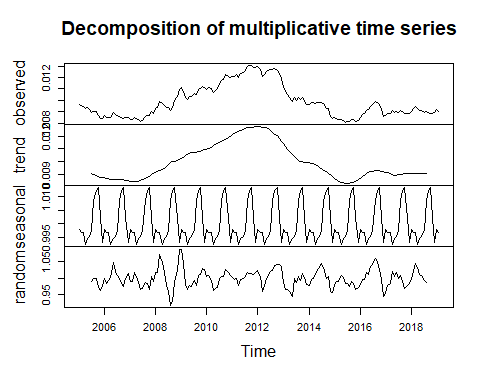
gold.decom.multiplicative <- decompose(gold, type = "mult")  
oil.decom.multiplicative <- decompose(oil, type = "mult")  
yen.decom.multiplicative <- decompose(yen, type = "mult")  
  
plot(gold.decom.multiplicative)



plot(oil.decom.multiplicative)



plot(yen.decom.multiplicative)



#Answer: in both cases multiplicative and additive show same trend, but multiplicative  
# had variation remain around 0 in mean for seasonal and random variables, so we should pick additive for that reason.  
  
  
  
#16 For each of the variables extract the random component and save   
# them as "goldrand", "oilrand", and "yenrand". Moreover, use na.omit()  
# function to deal with the missing values.  
  
goldrand = na.omit(gold.decom.additive$random)  
oilrand = na.omit(oil.decom.additive$random)  
yenrand = na.omit(yen.decom.additive$random)  
  
#17 For the random component of each of the assets, please estimate   
# autocorrelation function.Does any of the assets exhibit autocorrelation?  
# If yes, to what degree?  
# Keep in mind there are missing values.   
  
acf(goldrand)  
acf(oilrand)  
acf(yenrand)

Graphical user interface

Description automatically generated

pacf(oilrand)  
pacf(goldrand)  
pacf(yenrand)

Graphical user interface

Description automatically generated

#Answer: all of the values show autocrellation as we see a wave like pattern in a decreasing direction  
# OilRand shows highest ACF in the first lag when we look at pacf and is almost 0.8.  
  
  
#18 For all possible pairs of assets please estimate cross-correlation function   
# Do any of the variable lead or precede each other?  
# Could you use any of the varibales to predict values of other variables?  
# Make sure to use detrended and seasonally adjusted variables.   
# ("goldrand", "oilrand", and "yenrand")  
layout(matrix(1:6, nrow = 3, ncol = 2, byrow = TRUE))  
  
  
ts.plot(goldrand,oilrand,col=c(3,4))  
legend(x = "topright", # Position  
 legend = c("gold","oil"), # Legend texts  
 lty = c(4, 2), # Line types  
 col = c(3,4), # Line colors  
 lwd = 2) # Line width  
  
ccf(goldrand,oilrand,main="CCF of goldrand and oilrand", ylab = "", xlab = "", col = "blue", ci.col="red")  
  
#################  
  
ts.plot(goldrand,yenrand,col=c(3,4))  
legend(x = "topright", # Position  
 legend = c("gold","yen"), # Legend texts  
 lty = c(4, 2), # Line types  
 col = c(3,4), # Line colors  
 lwd = 2) # Line width  
  
ccf(goldrand,yenrand,main="CCF of goldrand and yenrand", ylab = "", xlab = "", col = "blue", ci.col="red")  
######  
  
ts.plot(yenrand,oilrand,col=c(3,4))  
legend(x = "topright", # Position  
 legend = c("yen","oil"), # Legend texts  
 lty = c(4, 2), # Line types  
 col = c(3,4), # Line colors  
 lwd = 2) # Line width  
  
ccf(yenrand,oilrand,main="CCF of yenrand and oilrand", ylab = "", xlab = "", col = "blue", ci.col="red")

Diagram

Description automatically generated

#Answer:variables dont lead or preceed much consistently, oil in most cases is lagging behind and yen seems to lead but incosistnently in lags.  
  
#19 Based on the time series plot of gold, oil, and yen prices,   
# there appears to be no systematic trends or seasonal effects.   
# Therefore, it is reasonable to use exponential smoothing for these time series.  
# Estimate alpha, the smoothing parameter for gold, oil and yen.   
# What does the value of alpha tell you tell you about the behavior of the mean?   
# What is the estimated value of the mean for each asset?  
  
gold\_smooth = HoltWinters(gold, beta = 0, gamma = 0)  
oil\_smooth = HoltWinters(oil, beta = 0, gamma = 0)  
yen\_smooth = HoltWinters(yen, beta = 0, gamma = 0)  
  
  
gold\_smooth

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = gold, beta = 0, gamma = 0)  
##   
## Smoothing parameters:  
## alpha: 0.9999196  
## beta : 0  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 1317.813562  
## b 14.375654  
## s1 -10.446046  
## s2 33.326755  
## s3 87.130633  
## s4 -3.000999  
## s5 -26.579571  
## s6 -24.278783  
## s7 -16.737978  
## s8 -15.299397  
## s9 -26.929144  
## s10 -11.735742  
## s11 12.610796  
## s12 1.939475

oil\_smooth

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = oil, beta = 0, gamma = 0)  
##   
## Smoothing parameters:  
## alpha: 0.9999373  
## beta : 0  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 60.4241433  
## b 0.8635184  
## s1 -4.4704514  
## s2 2.5357986  
## s3 3.8866319  
## s4 3.7537153  
## s5 0.2449653  
## s6 5.1678819  
## s7 4.8407986  
## s8 0.5891319  
## s9 -5.0542014  
## s10 -5.4162847  
## s11 -0.6033681  
## s12 -5.4746181

yen\_smooth

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = yen, beta = 0, gamma = 0)  
##   
## Smoothing parameters:  
## alpha: 0.9999335  
## beta : 0  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 9.188601e-03  
## b -4.127660e-05  
## s1 -5.538264e-05  
## s2 -9.677855e-06  
## s3 4.065366e-04  
## s4 1.737083e-04  
## s5 -7.941011e-05  
## s6 1.152514e-04  
## s7 1.479382e-04  
## s8 -6.280495e-05  
## s9 -2.754848e-04  
## s10 -2.384694e-04  
## s11 1.170997e-05  
## s12 -1.339148e-04

#Answer: After we smoothed gold, we see that gold has alpha of 0.9999196 (current observed value weight, meaning past observed values account for much lesser in terms of determining current value of estimate of the mean) and current estimate of the mean is coefficient a = 1317.813562, such high value for alpha means smoothing almost didnt happen and current value changed a lot.  
#Answer: After we smoothed oil, we see that gold has alpha of 0.9999373 (current observed value weight, meaning past observed values account for much lesser in terms of determining current value of estimate of the mean) and current estimate of the mean is coefficient a = 60.4241433, such high value for alpha means smoothing almost didnt happen and current value changed a lot.  
#Answer: After we smoothed yen, we see that gold has alpha of 0.9999335(current observed value weight, meaning past observed values account for much lesser in terms of determining current value of estimate of the mean) and current estimate of mean 9.188601e-03 is coefficient a, such high value for alpha means smoothing almost didnt happen and current value changed a lot.  
# ANswer: the value A in coefficient is the expected value for any numers of days ahead in the exponential smoothing  
  
#20 Use plot() function to generate three graphs that depict observed   
# and HoltWinter fitted values for each asset.  
  
plot(gold\_smooth)  
plot(oil\_smooth)  
plot(yen\_smooth)  
  
  
#21 Use window() function to create 3 new variables called   
# "goldpre", "oilpre", and "yenpre" that covers the period from January 2005,   
# until August 2018.   
  
goldpre = window(gold, start = c(2005,1), end = c(2018,8))   
oilpre = window(oil, start = c(2005,1), end = c(2018,8))   
yenpre = window(yen, start = c(2005,1), end = c(2018,8))   
  
#22 Use window() function to create 3 new variables called   
# goldpost, oilpost, and yenpost that cover the period from September 2018,   
# until February 2019.  
goldpost = window(gold, start = c(2018,9), end = c(2019,2))   
oilpost = window(oil, start = c(2018,9), end = c(2019,2))  
yenpost = window(yen, start = c(2018,9), end = c(2019,2))   
  
#23 Estimate HoltWinters filter model for each asset, while using only only pre data.  
# Save each of these estimates as "gold.hw", "oil.hw", and "yen.hw".  
gold.hw = HoltWinters(goldpre,seasonal = 'additive')  
gold.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = goldpre, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 0.8572867  
## beta : 0.03101722  
## gamma: 1  
##   
## Coefficients:  
## [,1]  
## a 1198.1930522  
## b -2.1675372  
## s1 2.5061057  
## s2 -26.8815961  
## s3 -38.8898693  
## s4 -51.7983177  
## s5 -5.7212105  
## s6 18.1013981  
## s7 8.2484540  
## s8 -0.5993937  
## s9 -3.5512328  
## s10 7.2344144  
## s11 -1.1028923  
## s12 3.6660387

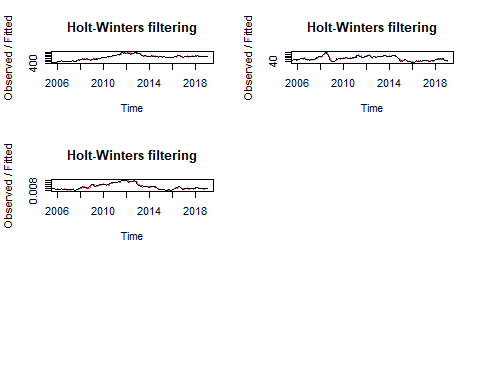
oil.hw = HoltWinters(oilpre,seasonal = 'additive')  
oil.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = oilpre, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 1  
## beta : 0.005599748  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 62.8921181  
## b 0.3491132  
## s1 4.8407986  
## s2 0.5891319  
## s3 -5.0542014  
## s4 -5.4162847  
## s5 -0.6033681  
## s6 -5.4746181  
## s7 -4.4704514  
## s8 2.5357986  
## s9 3.8866319  
## s10 3.7537153  
## s11 0.2449653  
## s12 5.1678819

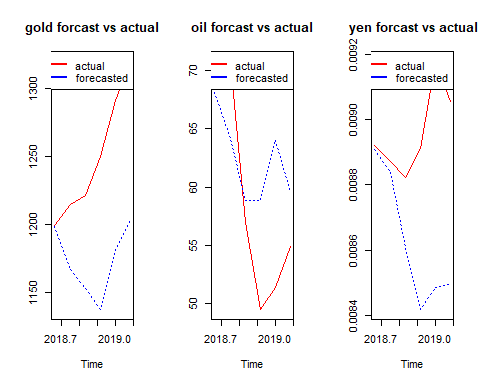
yen.hw = HoltWinters(yenpre,seasonal = 'additive')  
yen.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = yenpre, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 0.8612286  
## beta : 0.06008427  
## gamma: 1  
##   
## Coefficients:  
## [,1]  
## a 8.732109e-03  
## b -2.194822e-05  
## s1 1.981890e-04  
## s2 1.498968e-04  
## s3 -5.760429e-05  
## s4 -2.263830e-04  
## s5 -1.381907e-04  
## s6 -1.024081e-04  
## s7 -1.578987e-04  
## s8 -1.493823e-04  
## s9 -7.458267e-05  
## s10 1.494413e-04  
## s11 1.835743e-04  
## s12 2.771825e-04

#24 Use HoltWinters filter estimates generated in#23 and predict() function   
# to create a 6 month ahead forecast of the gold, oil, and yen prices.   
# Save these forcasted values as "goldforc", "oilforc", and "yenforc".  
  
goldforc <- predict(gold.hw, n.ahead = 6)  
oilforc <- predict(oil.hw, n.ahead = 6)  
yenforc <- predict(yen.hw, n.ahead = 6)  
  
  
#25 Use ts.plot() function to plot side-by-side post sample prices   
# ("goldpost", "oilpost","yenpost") and their forecasted counterparts.  
# Please designate red color to represent the actual prices,   
# and blue doted lines to represent forecasted values.   
par(mfrow=c(1,3))



ts.plot(goldpost,goldforc, main = 'gold forcast vs actual', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)  
  
ts.plot(oilpost,oilforc,main = 'oil forcast vs actual', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)  
  
  
ts.plot(yenpost,yenforc, main = ' yen forcast vs actual', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)



#26 Please calculate forecast mean percentage error for each assets forecasting model.   
# Which asset's forecasting model has the lowest mean percentage error?  
gold\_forecast\_mean\_pct\_err = mean(((goldpost - goldforc)/goldpost)\*100)  
gold\_forecast\_mean\_pct\_err

## [1] 5.98464

oil\_forecast\_mean\_pct\_err = mean(((oilpost - oilforc)/oilpost)\*100)  
oil\_forecast\_mean\_pct\_err

## [1] -7.141958

yen\_forecast\_mean\_pct\_err = mean(((yenpost - yenforc)/yenpost)\*100)  
yen\_forecast\_mean\_pct\_err

## [1] 3.700225

#Answer: oil has lowest mean percentage error at -7.14%  
  
#27 Use gold, oil, and yen variables to estimate Holt-Winters model  
# for each asset. Save these estimates as "goldc.hw", "oilc.hw", and "yenc.hw".  
  
goldc.hw = HoltWinters(gold,seasonal = 'additive')  
goldc.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = gold, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 0.8669606  
## beta : 0.02672962  
## gamma: 1  
##   
## Coefficients:  
## [,1]  
## a 1301.2249693  
## b 1.2655991  
## s1 6.8731289  
## s2 -1.5828018  
## s3 -0.9402348  
## s4 9.4777023  
## s5 -1.1000113  
## s6 1.5776487  
## s7 -0.4333405  
## s8 -22.3268311  
## s9 -35.6789419  
## s10 -44.6940824  
## s11 -4.9671725  
## s12 18.5300307

oilc.hw = HoltWinters(oil,seasonal = 'additive')  
oilc.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = oil, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 1  
## beta : 0.005985716  
## gamma: 0  
##   
## Coefficients:  
## [,1]  
## a 60.4246181  
## b 0.2991417  
## s1 -4.4704514  
## s2 2.5357986  
## s3 3.8866319  
## s4 3.7537153  
## s5 0.2449653  
## s6 5.1678819  
## s7 4.8407986  
## s8 0.5891319  
## s9 -5.0542014  
## s10 -5.4162847  
## s11 -0.6033681  
## s12 -5.4746181

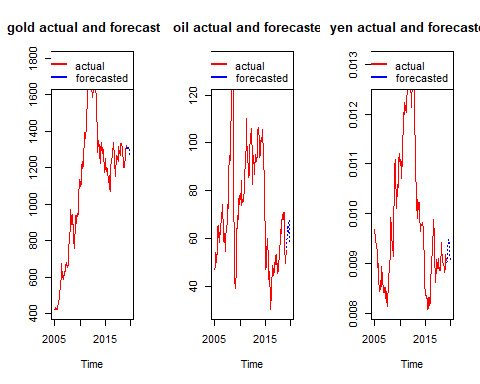
yenc.hw = HoltWinters(yen,seasonal = 'additive')  
yenc.hw

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = yen, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 0.864074  
## beta : 0.05782971  
## gamma: 1  
##   
## Coefficients:  
## [,1]  
## a 9.177452e-03  
## b 7.314582e-06  
## s1 -1.562923e-04  
## s2 -1.460248e-04  
## s3 -6.691235e-05  
## s4 1.553087e-04  
## s5 1.844854e-04  
## s6 2.735118e-04  
## s7 1.966265e-04  
## s8 1.502738e-04  
## s9 -3.638167e-05  
## s10 -1.890157e-04  
## s11 -1.106396e-04  
## s12 -1.227621e-04

#28 Use "goldc.hw", "oilc.hw", and "yenc.hw" models to create an out-of-sample  
# forecasts to predict the prices of each of the assets for the rest of the 2019.  
# Save these forecasts as "goldforcos", "oilforcos", "yenforcos".  
# What is the forecasted price of Gold for November 2019?   
  
goldforcos = predict(goldc.hw, n.ahead = 10)  
oilforcos = predict(oilc.hw, n.ahead = 10)  
yenforcos = predict(yenc.hw, n.ahead = 10)  
  
window(goldforcos, start = c(2019, 11), end = c(2019, 11))

## Nov  
## 2019 1276.936

#Answer: Gold forcasted price is 1276.936 for november 2019.  
  
# 29 Create time series plots for each asset, that combines the actual price data  
# of each asset and their out-of-sample forecasted values.  
# Please designate red color to represent the actual prices,   
# and blue doted lines to represent forecasted values.  
# What do you think will happen to the price of each asset by the end of the year?  
  
par(mfrow = c(1,3))  
ts.plot(gold, goldforcos, main = 'gold actual and forecasted', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)  
ts.plot(oil, oilforcos, main = 'oil actual and forecasted', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)  
ts.plot(yen, yenforcos, main = 'yen actual and forecasted', col = c('red','blue'),lty = c("solid", "dotted"))  
legend(x = "topright", legend = c("actual","forecasted"), col = c("red","blue"),lwd = 2)



#Answer: For gold price is expected to go down, for oil and yen there will be volatility but will end close to where actual values ended, so no change for oil and yen at the end of the year.  
  
  
# 30 Please calculate percentage change between the price of each asset in   
# February 2019 and their forecasted December 2019 prices.   
# Which asset promises the highest rate of return?

#ANSWER: Oil promises highest rate of return at 5.55%.

gold\_feb = window(gold, start = c(2019,2))[1]  
gold\_dec = goldforcos[10]  
gold\_percentage\_change = (gold\_dec - gold\_feb)/ gold\_feb  
gold\_percentage\_change \* 100

## [1] -3.831629

#formula for percentage change calculator( 100x ((final-initial)/initial))  
# Answer: Dec price will decrease by 3.83%.  
  
  
  
oil\_feb = window(oil, start = c(2019,2))[1]  
oil\_dec = oilforcos[10]  
oil\_percentage\_change = (oil\_dec - oil\_feb)/ oil\_feb  
oil\_percentage\_change \* 100

## [1] 5.550046

#formula for percentage change calculator( 100x ((final-initial)/initial))  
# Answer: Dec price will increase by 5.55%.  
  
  
yen\_feb = window(yen, start = c(2019,2), end = c(2019,2))[1]  
yen\_dec = window(yenforcos, start = c(2019,12), end = c(2019,12))[1]  
yen\_percentage\_change = (yen\_dec - yen\_feb)/ yen\_feb  
yen\_percentage\_change \* 100

## [1] 0.07611777

#formula for percentage change calculator( 100x ((final-initial)/initial))  
# Answer: Dec price will increase by 0.08%%.