###################################################################

###############################################################

# Project: Minimum Variance Portfolio

# Editors: Xinyuan Liu, Kevin Hu, Quinn Song, Tina Behroozi

# Date: November 30th 2019

###############################################################

###########################################

# Table of Contents

# 1. Data Preparation

# 2. Log-Return Calculation

# 3. Minimum Weight Optimization

# 4. ARIMA Model

# 5. Copula Model

# 6. Garch Model

###########################################

###########################################

## 1: DATA PREPARATION

###########################################

###########################################

## 1.1: Install Packages

###########################################

#install.packages("zoo")

#install.packages("devtools")

#install\_github("cvxgrp/CVXR")

#install.packages("CVXR")

#install.packages("tidyverse")

#install.packages("tseries")

#install.packages("forecast")

#install.packages("Metrics")

#install.packages("copula")

#install.packages("readxl")

#install.packages("MASS")

#install.packages("forecast")

#install.packages('rugarch')

#install.packages('Metrics')

###########################################

## 1.2: Call Libraries

###########################################

library(lubridate)

library(zoo)

library(devtools)

library(CVXR)

library(tidyverse)

library(tseries)

library(forecast)

library(Metrics)

library(copula)

library("readxl")

library("MASS")

library("forecast")

library('rugarch')

library('Metrics')

###########################################

## 1.3: Read Raw Dataset

###########################################

rawdata <- read.csv('C:/Users/hukevin/Desktop/Apps/energy\_data.csv')

###########################################

## 1.4: Imputation

###########################################

#  Carry the lastest observation forward to missing values

data<-na.locf(rawdata)

#  Use the mean of first six months to impute the first three days data

data[,3:14]<-as.numeric(unlist(data[,3:14]))

data$gas[1]<-mean(data$gas[4:120])

data$gas[2]<-mean(data$gas[4:120])

data$gas[3]<-mean(data$gas[4:120])

#  Check the missing consective data in xdr (our base asset)

missingxdr<-rawdata[which(is.na(rawdata$xdr)),]

data$Date<-as.Date(data$Date,format="%Y-%m-%d")

###########################################

## 1.5: Plot a few data for observation

###########################################

plot(x=data$Date,y=data$xdr)

plot(x=data$Date,y=data$cny)

###########################################

## 1.6: Convert all data from USD to XDR

###########################################

multiply <- function(x)

  {

  data[,x]\*data[,3]

  }

data[,4]<-data[,4]\*data[,3]

data[,5]<-multiply(5)

data[,6]<-multiply(6)

data[,7]<-multiply(7)

data[,8]<-multiply(8)

data[,9]<-multiply(9)

data[,10]<-multiply(10)

data[,11]<-multiply(11)

data[,12]<-multiply(12)

data[,13]<-multiply(13)

data[,14]<-multiply(14)

###########################################

## 2: LOG-RETURN CALCULATION

###########################################

###########################################

## 2.1: Calculate Log Return

###########################################

log\_returns<-data[-1,2]

# take a look at the log\_returns2 which is the xdr

log\_returns2<- diff(log(data[,3]), lag=1)

plot(log\_returns2)

log\_returns3<- diff(log(data[,4]), lag=1)

log\_returns4<- diff(log(data[,5]), lag=1)

log\_returns5<- diff(log(data[,6]), lag=1)

log\_returns6<- diff(log(data[,7]), lag=1)

log\_returns7<- diff(log(data[,8]), lag=1)

log\_returns8<- diff(log(data[,9]), lag=1)

log\_returns9<- diff(log(data[,10]), lag=1)

log\_returns10<- diff(log(data[,11]), lag=1)

log\_returns11<- diff(log(data[,12]), lag=1)

log\_returns12<- diff(log(data[,13]), lag=1)

log\_returns13<- diff(log(data[,14]), lag=1)

# Combine log returns data

newdata<-cbind(log\_returns,log\_returns2,log\_returns3,log\_returns4,

               log\_returns5,log\_returns6,log\_returns7,log\_returns8,

               log\_returns9,log\_returns10,log\_returns11,log\_returns12,

               log\_returns13)

newdata<-as.data.frame(newdata)

newdata$log\_returns<-data[-1,2]

# Assign columns name

colnames(newdata,do.NULL=FALSE)

colnames(newdata)<-c("DATE","XDR","EUR","JPY","CNY","BTC","ETH",

                     "RIP","MON","OIL","GAS","NAT","URA")

###########################################

## 2.2: Categorize data to each quarters

###########################################

newdata$Quarter <- ifelse(

  newdata$DATE< as.Date('2016-12-31'),0,

  ifelse(

    newdata$DATE < as.Date('2017-03-31'),1,

    ifelse(newdata$DATE< as.Date('2017-06-30'),2,

           ifelse(newdata$DATE < as.Date('2017-09-30'),3,

                  ifelse(newdata$DATE < as.Date('2017-12-31'),4,

                         ifelse(newdata$DATE < as.Date('2018-03-31'),5,

                                ifelse(newdata$DATE< as.Date('2018-06-30'),6,

                                       ifelse(newdata$DATE < as.Date('2018-09-30'),7,

                                              ifelse(newdata$DATE < as.Date('2018-12-31'),8,

                                                     ifelse(newdata$DATE< as.Date('2019-03-31'),9,

                                                            ifelse(newdata$DATE < as.Date('2019-06-30'),10,11)))))))))))

###########################################

## 3: Minimum Weight Optimization

###########################################

###########################################

## 3.1: Minimum Variance Weight Calculation

###########################################

# Initialize empty variance, covariance and weights

LogRetCov <- matrix(,12,1212)

LogRetVar <- matrix(,12,11)

optimw    <- matrix(,12,11)

# Loop the optimization model

for (i in 1:11)

  {

  z <- 12\*i+1

  k <- 12\*(i+1)

  # calculate covariance matrix

  LogRetCov[,z:k]<- cov(subset(newdata[,2:13],newdata$Quarter==i|newdata$Quarter==i-1))

  # calculate the variance

  LogRetVar\_1<-sapply(subset(newdata[,2:13],newdata$Quarter==i|newdata$Quarter==i-1),var)

  dim(LogRetVar\_1)<-c(12,1)

  LogRetVar[,i]<-LogRetVar\_1

  # quadration programming for weight optimization

  w <- Variable(12)

  objective <- Minimize(LogRetVar[,i] %\*% w + quad\_form(w,LogRetCov[,z:k]))

  constraints <- list(sum(w) == 1, w >= 0.01, w <= 0.25)

  prob <- Problem(objective, constraints)

  # output the weight

  result <- solve(prob)

  optimw[,i]<- result$getValue(w)

}

###########################################

## 3.2: Tidying Up the Results

###########################################

# Tidy-up the results

colnames(optimw,do.NULL=FALSE)

colnames(optimw)<-c("Q1","Q2","Q3","Q4","Q5","Q6","Q7",

                    "Q8","Q9","Q10","Q11")

# Label quarters on the results

colnames(data)[colnames(data) == "Date"] <- "DATE"

data$Quarter <- ifelse(data$DATE < as.Date('2017-03-31'),0,

                       ifelse(data$DATE< as.Date('2017-06-30'),1,

                              ifelse(data$DATE < as.Date('2017-09-30'),2,

                                     ifelse(data$DATE < as.Date('2017-12-31'),3,

                                            ifelse(data$DATE < as.Date('2018-03-31'),4,

                                                   ifelse(data$DATE< as.Date('2018-06-30'),5,

                                                          ifelse(data$DATE < as.Date('2018-09-30'),6,

                                                                 ifelse(data$DATE < as.Date('2018-12-31'),7,

                                                                        ifelse(data$DATE< as.Date('2019-03-31'),8,

                                                                               ifelse(data$DATE < as.Date('2019-06-30'),9,10))))))))))

###########################################

## 3.3: Calculating Quarter-End Results

###########################################

data$Quarter <- ifelse(data$DATE < as.Date('2017-03-31'),0,

                       ifelse(data$DATE< as.Date('2017-06-30'),1,

                              ifelse(data$DATE < as.Date('2017-09-30'),2,

                                     ifelse(data$DATE < as.Date('2017-12-31'),3,

                                            ifelse(data$DATE < as.Date('2018-03-31'),4,

                                                   ifelse(data$DATE< as.Date('2018-06-30'),5,

                                                          ifelse(data$DATE < as.Date('2018-09-30'),6,

                                                                 ifelse(data$DATE < as.Date('2018-12-31'),7,

                                                                        ifelse(data$DATE< as.Date('2019-03-31'),8,

                                                                               ifelse(data$DATE < as.Date('2019-06-30'),9,

                                                                                      ifelse(data$DATE==as.Date('2019-09-30'),10,9)))))))))))

data <- data[order(data$Quarter,data$DATE, decreasing=FALSE),]

data1<-subset(data,data$Quarter!=0)

#calculate the weighted index

data$Quarter <- ifelse(

  data$DATE< as.Date('2016-12-31'),0,

  ifelse(

    data$DATE < as.Date('2017-03-31'),1,

    ifelse(data$DATE< as.Date('2017-06-30'),2,

           ifelse(data$DATE < as.Date('2017-09-30'),3,

                  ifelse(data$DATE < as.Date('2017-12-31'),4,

                         ifelse(data$DATE < as.Date('2018-03-31'),5,

                                ifelse(data$DATE< as.Date('2018-06-30'),6,

                                       ifelse(data$DATE < as.Date('2018-09-30'),7,

                                              ifelse(data$DATE < as.Date('2018-12-31'),8,

                                                     ifelse(data$DATE< as.Date('2019-03-31'),9,

                                                            ifelse(data$DATE < as.Date('2019-06-30'),10,

                                                                   ifelse(data$DATE==as.Date('2019-09-30'),11,10))))))))))))

#quarter\_end price

data <- data[order(data$Quarter,data$DATE, decreasing=TRUE),]

qe\_price<-data[!duplicated(data$Quarter),]

#quarter\_end\_index

qe\_price<-qe\_price[order(qe\_price$Quarter,decreasing=FALSE),]

matrix1<-qe\_price[-1,-c(1,2,15)]

matrix2<-optimw

matrix3<-as.matrix(matrix1)%\*%as.matrix(matrix2)

qe\_index<-cbind(matrix3[1,1],matrix3[2,2],matrix3[3,3],matrix3[4,4],

                matrix3[5,5],matrix3[6,6],matrix3[7,7],matrix3[8,8],

                matrix3[9,9],matrix3[10,10],matrix3[11,11])

###########################################

## 3.4: Calculate Realized Index

###########################################

qe\_index2 <- as.data.frame(lapply(qe\_index, rep,12))

colnames(qe\_index2) <- c("Q1","Q2","Q3","Q4","Q5","Q6","Q7","Q8","Q9","Q10","Q11")

matrix1t <- data.frame(t(matrix1))

c <- qe\_index2/matrix1t

# Append weights for each quater

append<-cbind(data1,t(c[,data1[,15]]))

index<-rowSums(append[,3:14]\*append[,16:27])

index2<-c(rep(index[1],length(index)))

# Index Calculation

index<-index/index2\*100

cbind(append[,c(2,15)],index)

# Plot the index

plot(index)

# Label the index with dates

index3 <- cbind(append[,c(2,15)],index)

plot(index3[,1],index3[,3])

#correlation matrix of assets value

corr<-newdata[,c(-1,-14)]

source("http://www.sthda.com/upload/rquery\_cormat.r")

corrplot<-rquery.cormat(corr)

###########################################

## 4: ARIMA MODEL

###########################################

###########################################

## 4.1: Fitting ARIMA Model

###########################################

cat("\014")

rm(list = ls())

graphics.off()

# in this case, we have 9 quarters

index3 <- subset(index3,index3$Quarter<=10)

ARIMARMSE <- vector()

ARIMAMAPE <-vector()

ARIMAMSE <-vector()

for (i in 1:9)

{

IndexRealized <- index3 # Rename the realized index dataframe

TrainIndexi <- subset(IndexRealized, IndexRealized$Quarter <= i) # Our current training set

TrainIndexi <- TrainIndexi$index # Strip out date columns

TestIndexi <- subset(IndexRealized, IndexRealized$Quarter == i+1) # Our current testing set

TestIndexi <- TestIndexi$index # Strip out date columns

ARIMAi <- auto.arima(TrainIndexi

,max.p = 10

,max.q = 5

,ic = "aic") # Fit ARIMA Model

show(ARIMAi)

Forecasti <- forecast(ARIMAi, h = length(TestIndexi),bootstrap = T)

meanboot = Forecasti$mean

RMSEi <- rmse(meanboot, TestIndexi)

ARIMARMSE <- cbind(ARIMARMSE, RMSEi)

MAPEi <- mape(meanboot, TestIndexi)

ARIMAMAPE <- cbind(ARIMAMAPE, MAPEi)

MSEi <- mse(meanboot, TestIndexi)

ARIMAMSE <- cbind(ARIMAMSE, MSEi)

}

ARIMARMSE

ARIMAMAPE

ARIMAMSE

measure <- rbind(ARIMARMSE,ARIMAMAPE,ARIMAMSE)

rownames(measure)<-c('rmse','mape','mse')

measuret <- t(measure)

###########################################

## 4.2: Backtesting ARIMA Model

###########################################

TrainIndex <- subset(index3, index3$Quarter <= 7) # Our current training set

TrainIndex <- TrainIndex$index # Strip out date columns

TestIndex <- subset(index3, index3$Quarter > 7) # Our current testing set

TestIndex <- TestIndex$index # Strip out date columns

ARIMA <- auto.arima(TrainIndex

,max.p = 10

,max.q = 5

,ic = "aic") # Fit ARIMA Model

Forecasti <- forecast(ARIMA, h = length(TestIndex)+10,bootstrap = T)

meanboot = Forecasti$mean

upperboot = Forecasti$upper[,1]

lowerboot = Forecasti$lower[,1]

xdr = Index2$xdr

btc = Index2$btc

gas = Index2$gas

q = as.character(index3$Quarter)

q[1] = '17/1'

q[277] = '18/4'

q[642] = '19/8'

a = c(1,which.max(index3$Quarter==2),which.max(index3$Quarter==3),277,which.max(index3$Quarter==5),

which.max(index3$Quarter==6),which.max(index3$Quarter==7),642,which.max(index3$Quarter==9),which.max(index3$Quarter==10))

plot(index3$index, xaxt = "n",xlab='year/quarter',ylab='Index',main='ARIMA model')

axis(1, at=a, labels=q[a])

lines(c(fitted(Forecasti),upperboot),col='blue')

lines(c(fitted(Forecasti),lowerboot),col='green')

lines(c(fitted(Forecasti),meanboot),col='red')

plot(c(fitted(Forecasti),meanboot), xaxt = "n",xlab='year/quarter',ylab='Index',main='ARIMA model',col='black')

axis(1, at=a, labels=q[a])

lines(gas,col='red', lty=2)

lines(xdr,col='blue', lty=3)

lines(btc,col='green', lty=4)

legend("topleft", legend=c('index',"gas", "xdr",'btc'),

col=c('black',"red", "blue",'green'), lty=1:4, cex=0.8)

##ACF and PACF

kpss.test(TrainIndex)

kpss.test(diff(TrainIndex))

kpss.test(diff(diff(TrainIndex)))

Acf(diff(diff(TrainIndex)))

Pacf(diff(diff(TrainIndex)))

MAresult = arma(diff(diff(TrainIndex)),order = c(0, 1))

MAresult = as.numeric(na.omit(MAresult$fitted.values))

Acf(MAresult)

Pacf(MAresult)

###########################################

## 5: Copula Simulation of Index

###########################################

# Simulates index using copula:

# We use previous month’s data to create empirical marginal cdfs

# and a copula which we can combine to simulate more data.

# Together with the weights obtained through our previous optimization,

# we can simulate the index for the next month

library(copula)

library(sn)

#library(ks)

set.seed(999)

source('FINAL DATA IMPUTE AND CLEANUP CODE.R')#this is index3

# container for simulated data

sim\_data\_guass\_cop = data.frame(array(NA,c(sum(my\_data\_3\_xdr$quarters>0),14)))

names(sim\_data\_guass\_cop) = names(my\_data\_3\_xdr)

# populate with correct quarters

sim\_data\_guass\_cop$quarters=my\_data\_3\_xdr$quarters[my\_data\_3\_xdr$quarters>0]

# populate with correct dates

sim\_data\_guass\_cop$Date=my\_data\_3\_xdr$Date[my\_data\_3\_xdr$quarters>0]

for(i in 1:11){

# convert log diffs to empirical cdf of log diffs

# get stationary values

x = my\_data\_4\_log\_diff[my\_data\_4\_log\_diff$quarters==i-1,c(-1,-dim(my\_data\_4\_log\_diff)[2])]

#print(head(u))

#print(dim(u)[1])

# get u's which are empiracle observations of log diff distribution

u = x

for(j in 1:dim(u)[2]){

u[,j] = pobs(x[,j])

}

u = as.matrix(u)

# now fit copula

## Inverting Kendall's tau

fit.tau <- fitCopula(normalCopula(dim=12, dispstr="un"), u, method="itau")

# get number of log diffs we need to simulate

n = sum(my\_data\_3\_xdr$quarters==i)

# simulate

sim\_log\_diffs = rCopula(n, fit.tau@copula)

#print(n)

# change from uniform back to draws from log diff distributions

# through using quantiles (cdf approximation)

for(j in 1:dim(sim\_log\_diffs)[1]){

for(k in 1:dim(sim\_log\_diffs)[2]){

sim\_log\_diffs[j,k] = quantile(x[,k],sim\_log\_diffs[j,k])

}

}

#print(sim\_log\_diffs[1:3])

# index of last date in training set (need this to go back from stationary to (simulated) price)

ind = which(max(my\_data\_3\_xdr$Date[my\_data\_3\_xdr$quarters==i-1])==my\_data\_3\_xdr$Date)

#print(ind)

# now get back simulated price! complicated forumla

# initialize return matrix

sim\_prices = sim\_log\_diffs

for(j in 1:dim(sim\_log\_diffs)[2]){

sim\_prices[,j] = exp(cumsum(sim\_log\_diffs[,j])+log(my\_data\_3\_xdr[ind,1+j]))

}

# add back to matrix!!!

sim\_data\_guass\_cop[sim\_data\_guass\_cop$quarters==i,c(-1,-dim(sim\_data\_guass\_cop)[2])] = sim\_prices

}

#head(sim\_prices)

# get weights

weights = read.csv('realizedweights.csv')

# add to model

cop\_sim\_with\_weights = cbind(sim\_data\_guass\_cop,

data.frame(t(weights[,-1][,sim\_data\_guass\_cop$quarters])))

# dont let rownames be weird

rownames(cop\_sim\_with\_weights) = 1:(dim(cop\_sim\_with\_weights)[1])

# check

#cop\_sim\_with\_weights[80:100,]

#ADD COP INDEX

cop\_sim\_with\_weights['cop\_pred\_index'] = 1

#rowSums(cop\_sim\_with\_weights[,2:(2+11)] \* cop\_sim\_with\_weights[,15:(15+11)])

for(i in 1:11){

last\_date\_prev\_quarter = max(my\_data\_3\_xdr$Date[my\_data\_3\_xdr$quarters==i-1])

# ind is last date before current quarter from original data

ind = (which(my\_data\_3\_xdr$Date==last\_date\_prev\_quarter))

# ind2 is last date before current quarter from cop simulated data

ind2 = (which(cop\_sim\_with\_weights$Date==last\_date\_prev\_quarter))

if(length(ind2)==0){

last\_val = 100

} else{

last\_val = cop\_sim\_with\_weights$cop\_pred\_index[ind2]

}

# make cop\_index

# count how many elements in this month

n = sum(cop\_sim\_with\_weights$quarters==i)

# get the subset of the data frame with only this month

temp = cop\_sim\_with\_weights[cop\_sim\_with\_weights$quarters==i,]

# index value!!! calculated with formula

cop\_sim\_with\_weights$cop\_pred\_index[cop\_sim\_with\_weights$quarters==i] =

last\_val\*rowSums(temp[,2:(2+11)] \* temp[,15:(15+11)] /

(as.matrix(array(rep(1,n),c(n,1)))%\*%

as.matrix(array(my\_data\_3\_xdr[ind,2:13],c(1,12)))))

}

# plot comparing cop index to bitcoin

btc = my\_data\_3\_xdr[my\_data\_3\_xdr$quarters>0,]$btc

btc = (btc-mean(btc))/sqrt(var(btc))

plot(cop\_sim\_with\_weights$Date,btc,col="blue",type='l')

points(cop\_sim\_with\_weights$Date,(cop\_sim\_with\_weights$cop\_pred\_index-mean(cop\_sim\_with\_weights$cop\_pred\_index))/

sqrt(var(cop\_sim\_with\_weights$cop\_pred\_index)),col='red',type='l')

points(cop\_sim\_with\_weights$Date,10000\*(0.5-c(0,diff(cop\_sim\_with\_weights$quarters))),col="black",type='o')

# var of the index

var(cop\_sim\_with\_weights$cop\_pred\_index)

# plot comparing bitcoin price simulated by copula to actual price

plot(cop\_sim\_with\_weights$Date,cop\_sim\_with\_weights$btc,col='red',type='l')

points(cop\_sim\_with\_weights$Date,my\_data\_3\_xdr[my\_data\_3\_xdr$quarters>0,]$btc,col="blue",type='l')

points(cop\_sim\_with\_weights$Date,10000\*(-c(0,diff(cop\_sim\_with\_weights$quarters))),col="black",type='o')

###########################################

## 5: Garch Model

###########################################

cat("\014")

rm(list = ls())

graphics.off()

index3 <- subset(Index3,Index3$Quarter<=10)

q = as.character(index3$Quarter)

q[1] = '17/1'

q[277] = '18/4'

q[642] = '19/8'

a = c(1,which.max(index3$Quarter==2),which.max(index3$Quarter==3),277,which.max(index3$Quarter==5),

which.max(index3$Quarter==6),which.max(index3$Quarter==7),642,which.max(index3$Quarter==9),which.max(index3$Quarter==10))

#index property

Data = index3$index

kpss.test(Data)

log.Data = log(Data)

kpss.test(log.Data)

sqrt.Data = sqrt(Data)

kpss.test(sqrt.Data)

boxcox(Data~1)

bc = boxcox(Data~1, lambda = seq(1.5, 2, 1/100), interp = FALSE)

lamba = 1.95

y = (Data^(lamba)-1)/lamba

kpss.test(y)

diff.Data = diff(Data)

kpss.test(diff.Data)

diff.diff.Data = diff(diff(Data))

kpss.test(diff.diff.Data)

par(mfrow=c(1,2))

plot(Data, xaxt = "n",xlab='year/quarter',ylab='Index')

axis(1, at=a, labels=q[a])

plot(diff.diff.Data, xaxt = "n",xlab='year/quarter',ylab='transform Index')

axis(1, at=a, labels=q[a])

par(mfrow=c(1,1))

boxplot(diff.diff.Data, main = "boxplot of diff(diff(Index))")

plot(density(diff.diff.Data), main = "kernel density plot of diff(diff(Index))")

tdist\_par = fitdistr(diff.diff.Data, "t", start = list(m=mean(diff.diff.Data),s=sd(diff.diff.Data), df=3), lower=c(-1, 0.001,1))

tdist\_ind = tdist\_par$estimate[2]\*rt(length(diff.diff.Data),df=tdist\_par$estimate[3]) + tdist\_par$estimate[1]

par(mfrow=c(1,2))

qqnorm(diff.diff.Data,datax = TRUE, main = "normal Q-Q plot")

qqplot(diff.diff.Data, tdist\_ind, main = "tdist Q-Q plot")

#fitting model

GARCHRMSE <- vector()

GARCHMAPE <-vector()

GARCHMSE <-vector()

APARCHRMSE <- vector()

APARCHMAPE <-vector()

APARCHMSE <-vector()

for (i in 1:2)

{

IndexRealized <- index3 # Rename the realized index dataframe

TrainIndexi <- subset(IndexRealized, IndexRealized$Quarter <= i) # Our current training set

TrainIndexi <- TrainIndexi$index # Strip out date columns

TestIndexi <- subset(IndexRealized, IndexRealized$Quarter == i+1) # Our current testing set

TestIndexi <- TestIndexi$index # Strip out date columns

ARIMAi <- auto.arima(TrainIndexi

,max.p = 10

,max.q = 5

,trace = TRUE

,ic = "aic") # Fit ARIMA Model

Armavar = ARIMAi$arma

if(Armavar[6]==1){

TrainIndexi = diff(TrainIndexi)

TestIndexi = diff(TestIndexi)

} else if(Armavar[6]==2){

TrainIndexi = diff(diff(TrainIndexi))

TestIndexi = diff(diff(TestIndexi))

} else if(Armavar[6]==3){

TrainIndexi = diff(diff(diff(TrainIndexi)))

TestIndexi = diff(diff(diff(TestIndexi)))

}

specgarch = ugarchspec(mean.model = list(armaOrder = c(Armavar[1],Armavar[2]), include.mean = TRUE), distribution.model = "std")

fit = ugarchfit(specgarch, data = TrainIndexi)

bootpred = ugarchboot(fit, method = "Partial", n.ahead = length(TestIndexi), n.bootpred = 1000)

u = bootpred@forc@forecast

meanboot = u$seriesFor

specaparch = ugarchspec(mean.model = list(armaOrder = c(Armavar[1],Armavar[2]), include.mean = TRUE),

variance.model = list(model = "apARCH"), distribution.model = "std")

fit = ugarchfit(specaparch, data = TrainIndexi)

show(fit)

bootpredaparch = ugarchboot(fit, method = "Partial", n.ahead = length(TestIndexi), n.bootpred = 1000)

u = bootpredaparch@forc@forecast

meanbootaparch = u$seriesFor

RMSEi <- rmse(meanboot, TestIndexi)

GARCHRMSE <- cbind(GARCHRMSE, RMSEi)

RMSEi <- rmse(meanbootaparch, TestIndexi)

APARCHRMSE <- cbind(APARCHRMSE, RMSEi)

MAPEi <- mape(meanboot, TestIndexi)

GARCHMAPE <- cbind(GARCHMAPE, MAPEi)

MAPEi <- mape(meanbootaparch, TestIndexi)

APARCHMAPE <- cbind(APARCHMAPE, MAPEi)

MSEi <- mse(meanboot, TestIndexi)

GARCHMSE <- cbind(GARCHMSE, MSEi)

MSEi <- mse(meanbootaparch, TestIndexi)

APARCHMSE <- cbind(APARCHMSE, MSEi)

}

GARCHRMSE

APARCHRMSE

GARCHMAPE

APARCHMAPE

GARCHMSE

APARCHMSE

#backtesting: put seven quarters as historical data

TrainIndex <- subset(index3, IndexRealized$Quarter <= 5)

TrainIndex <- TrainIndex$index

TestIndex <- subset(index3, IndexRealized$Quarter > 5)

TestIndex <- TestIndex$index

specgarch = ugarchspec(mean.model = list(armaOrder = c(3,0), include.mean = TRUE), distribution.model = "std")

fit = ugarchfit(specgarch, data = (diff(TrainIndex)))

bootpred = ugarchboot(fit, method = "Partial", n.ahead = (length(TestIndex)-1),

n.bootpred = 1000)

u = bootpred@forc@forecast

meanboot = u$seriesFor

zr = t(as.data.frame(bootpred, which = "series", type = "summary"))

upperboot = zr[,2]

lowerboot = zr[,4]

fitdata = ((fit@fit$fitted.values))

indexp = diffinv(c(diff((TrainIndex)),meanboot),xi = 100)

indexpu = diffinv(c(diff((TrainIndex)),upperboot),xi = 100)

indexpl = diffinv(c(diff((TrainIndex)),lowerboot),xi = 100)

par(mfrow=c(1,1))

plot(diff(index3$index), xaxt = "n",xlab='year/quarter',ylab='Index',main='GARCH model for diff(index)')

axis(1, at=a, labels=q[a])

lines(c(fitdata,meanboot),col='red')

lines(c(fitdata,upperboot),col='blue')

lines(c(fitdata,lowerboot),col='green')

plot((index3$index), xaxt = "n",xlab='year/quarter',ylab='Index',main='GARCH model for index')

axis(1, at=a, labels=q[a])

lines(indexp,col='red')

lines(indexpu,col='blue')

lines(indexpl,col='green')

plot(fit,which=8)

Res <- residuals(fit, standardize = TRUE)

skewness(Res) #left skew

kurtosis(Res) #thick tail

shapiro.test(as.numeric(Res))

specaparch = ugarchspec(mean.model = list(armaOrder = c(3,0), include.mean = TRUE),

variance.model = list(model = "apARCH"), distribution.model = "std")

fit = ugarchfit(specaparch, data = (diff(TrainIndex)))

bootpredaparch = ugarchboot(fit, method = "Partial", n.ahead = (length(TestIndex)-1), n.bootpred = 1000)

u = bootpredaparch@forc@forecast

meanbootaparch = u$seriesFor

zr = t(as.data.frame(bootpredaparch, which = "series", type = "summary"))

upperbootaparch = zr[,2]

lowerbootaparch = zr[,4]

fitdata = fit@fit$fitted.values

indexpaparch = diffinv((c(diff((TrainIndex)),meanbootaparch)),xi = 100)

indexpuaparch = diffinv((c(diff((TrainIndex)),upperbootaparch)),xi = 100)

indexplaparch = diffinv((c(diff((TrainIndex)),lowerbootaparch)),xi = 100)

plot(diff(index3$index), xaxt = "n",xlab='year/quarter',ylab='Index',main='APARCH model for diff(index)')

axis(1, at=a, labels=q[a])

lines(c(fitdata,meanbootaparch),col='red')

lines(c(fitdata,upperbootaparch),col='blue')

lines(c(fitdata,lowerbootaparch),col='green')

plot((index3$index), xaxt = "n",xlab='year/quarter',ylab='Index',main='APARCH model for index')

axis(1, at=a, labels=q[a])

lines(indexpaparch,col='red')

lines(indexpuaparch,col='blue')

lines(indexplaparch,col='green')

plot(fit,which=8)

Res <- residuals(fit, standardize = TRUE)

skewness(Res) #left skew

kurtosis(Res) #thick tail

shapiro.test(as.numeric(Res))