```r
#################### Financial Econometrics - Spring 2021 ######################
#################### Karoll Gómez Portilla ######################
#################### Universidad Nacional de Colombia ######################
#################### Lab 4: Unconditional volatility measures ######################
library(xts)
library(quantmod) # package has a function to obtain data from various sources
library(TTR)      # Technical Trading Rules
install.packages("FinTS", repos="http://R-Forge.R-project.org")
install.packages("USlobbying", repos="http://R-Forge.R-project.org")
install.packages("tstPkg1", repos="http://R-Forge.R-project.org")
install.packages("tstPkg2", repos="http://R-Forge.R-project.org")
library(stats)
library(greybox)
library(smooth)

getSymbols("SPY", from = "2000-01-03", to = "2013-12-31", src = "yahoo", adjust = TRUE)
head(SPY) # Open, high, low, close, volume and ADJUSTED price
# Plot the closing prices of SPY
plot(Cl(SPY))

# Add a 200-day SMA using lines()
lines(SMA(Cl(SPY), n = 200), col = "red")
lines(EMA(Cl(SPY), n = 200), col = "green")

# Returns
ret = diff(log(Cl(SPY)), lag=1)
plot(ret)

# Add a 200-day SMA using lines()
lines(SMA(ret, n = 100), col = "red")
lines(EMA(ret, n = 100), col = "green")


library(PerformanceAnalytics)
# Compute the rolling 1 month estimate of annualized volatility
chart.RollingPerformance(ret, width = 22, FUN = "sd.annualized", scale = 252, main = "One month rolling volatility")

# Compute the rolling 3 months estimate of annualized volatility
chart.RollingPerformance(ret, width = 66, FUN = "sd.annualized", scale = 252, main = "Three months rolling volatility")



### Note: highfrequency package allows to manage highfrequency trades and quotes data, calculate various liquidity measures, estimate and forecast volatility, detect price jumps and investigate microstructure
 noise and intraday periodicity.

#################### Financial Econometrics - Fall 2018 ######################
#################### Karoll Gómez Portilla ######################
#################### Master in Economics - Universidad Nacional de Colombia ######################
#################### Lab 5: Condictional volatility measures ######################

##### GARCH Exercise 1
# Q1. Load the rugarch package and the dmbp dataset (Bollerslev, T. and Ghysels, E. 1996, Periodic Autoregressive Conditional Heteroscedasticity, Journal of Business and Economic Statistics, 14, 139–151).
 This dataset has daily logarithmic nominal returns for Deutsche-mark / Pound. There is also a dummy variable to indicate non-trading days.
library(rugarch)
data("dmbp")
# Q2: Define the daily return as a time series variable and plot the return against time.
rets <- ts(dmbp$V1)
plot(rets)
# Q3:Plot the graph of the autocorrelation function of returns.
acf(rets)

# Q4: Plot the graph of the autocorrelation function of squared returns
acf(rets^2)

# Q5: We will to simulate and analyze an ARCH process.
# Use the ugarchspec function to define an ARCH(1) process. The return has a simple mean specification with mean=0. The variance follows as AR-1 process with constant=0.2 and AR-1 coefficient = 0.7.
arch1.mod = ugarchspec(variance.model = list(garchOrder=c(1,0)), mean.model = list(armaOrder=c(0,0)), fixed.pars=list(mu=0, omega=0.2, alpha1=0.7))
arch1.mod

# Q6: Simulate the ARCH process.
arch1.sim = ugarchpath(arch1.mod,n.sim=500)

# Q7: Plot the returns vs time and note the apparent unpredictability. Plot the path of conditional sigma vs time and note that there is some persistence over time.
plot(arch1.sim, which=2) # Returns
plot(arch1.sim, which=1) # Volatility

# Q8: Plot the ACF of returns and squared returns.
acf(arch1.sim@path$seriesSim, main="returns")
acf(arch1.sim@path$seriesSim^2, main="squared returns")


# Q9: Test for ARCH effects using the Ljung Box test for the simulated data and currency returns data.
Box.test(arch1.sim@path$seriesSim^2, type = "Ljung-Box", lag = 12)
Box.test(rets^2, type = "Ljung-Box", lag = 12)



##### GARCH Exercise 2
# Q2. We will first simulate and analyze a GARCH process. Use the ugarchspec function to define an GARCH(1,1) process. The return has a simple mean specification with mean=0. The variance follows as AR-1
 process with constant=0.2, AR-1 coefficient = 0.2 and MA-1 coefficient = 0.6.
garch1.mod = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder=c(0,0)), fixed.pars=list(mu=0, omega=0.2, alpha1=0.2, beta1=0.75))
garch1.mod

# Q3: Simulate the defined GARCH process for 500 time periods.
garch1.sim = ugarchpath(garch1.mod,n.sim=500)

# Q4: Plot the returns vs time and note the apparent unpredictability. Plot the path of conditional sigma vs time and note that there is some persistence over time.
plot(garch1.sim, which=2)
plot(garch1.sim, which=1)

# Q5: Plot the ACF of returns and squared returns.
acf(garch1.sim@path$seriesSim, main="returns")
acf(garch1.sim@path$seriesSim^2, main="squared returns")

# Q6: Test for ARCH effects using the Ljung Box test for the simulated data.
Box.test(garch1.sim@path$seriesSim^2, type = "Ljung-Box", lag = 12)

# Q7: We will now use the currency returns data (rets). Use the ugarchfit function to estimate a GARCH(1,1) model for the data. The return has a simple mean specification with mean=0.
garch11.spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder=c(0,0)))
dmbp.garch11.fit =ugarchfit(spec=garch11.spec, data=rets)
dmbp.garch11.fit

# H0 : No serial correlation (Q-statistics) This is testing the null hypothesis of adequately fitted ARCH process
# Nyblon test: H0:constancy of all parameters, ie, no structural change.
# Sign Bias Test: H0: leverage effect (positive or negative coefficents are statisticaly significant)
# The chi-squared goodness of fit test, compares the empirical distribution of the standardized residuals with the theoretical ones from the chosen density (normal in this case).


# Q8. Plot the fit diagnostics graphs.
plot(dmbp.garch11.fit,which="all")

# Q9: Fit an AR(1)-GARCH(1,1) model to the data.
garch11.spec = ugarchspec(variance.model = list(garchOrder=c(1,1)), mean.model = list(armaOrder=c(1,0)))
dmbp.garch11.fit = ugarchfit(spec=garch11.spec,data=rets)
dmbp.garch11.fit

# Q10: Plot the fit diagnostics graphs for the new model.
```

```
plot(dmbp.garch11.fit,which="all")



##### GARCH Exercise 3
# Q1: Load the rugarch and the FinTS packages. Next, load the m.ibmspln dataset from the FinTS package. This dataset contains monthly excess returns of the S&P500 index from Jan-1926 to Dec-1999 (Ruey Tsay
 (2005) Analysis of Financial Time Series, 2nd ed.,Wiley, chapter 3). Also, load the forecast package which we will use for auto-correlation graphs.
library(zoo)
library(rugarch)
library(FinTS)
install.packages("FinTS", repos="http://R-Forge.R-project.org")
install.packages("USlobbying", repos="http://R-Forge.R-project.org")
install.packages("tstPkg1", repos="http://R-Forge.R-project.org")
install.packages("tstPkg2", repos="http://R-Forge.R-project.org")
library(forecast)
data(m.ibmspln)

# Q2: Excess S&P500 returns are defined as a regular zoo variable. Convert this to a time series variable with correct dates.
sp <- as.ts(m.ibmspln)
sp <- sp[,2]
sp <- ts(sp, start = c(1926,1), end = c(1999,12), frequency = 12)

# Q3: Plot the excess S&P500 returns along with its ACF and PACF graphs and comment on the apparent correlation.
plot(sp)
Acf(sp, lag.max = 24)
Pacf(sp, lag.max = 24)

# Q4: Plot the squared excess S&P500 returns along with its ACF and PACF graphs and comment on the apparent correlation.
plot(sp^2)
Acf(sp^2, lag.max = 24)
Pacf(sp^2, lag.max = 24)

# Q5: Using the results from Q3, estimate a suitable ARMA model for excess returns assuming normal errors.
ar3 <- Arima(sp, order = c(3,0,0), include.constant = TRUE)
ar3

# Q6: Using the results from Q4, estimate a suitable ARMA model for excess returns without assuming normal errors
ar3garch11 = ugarchspec(variance.model = list(garchOrder=c(1,1)),
                          mean.model = list(armaOrder=c(3,0)))
ar3garch11.fit = ugarchfit(spec=ar3garch11,data=sp)
ar3garch11.fit

# Q7: Using the results from Q5 and Q6, estimate a more parsimonious model that has better fit
ar0garch11 = ugarchspec(variance.model = list(garchOrder=c(1,1)),
                          mean.model = list(armaOrder=c(0,0)))
ar0garch11.fit = ugarchfit(spec=ar0garch11,data=sp)
ar0garch11.fit

# Q8: Generate 10 steps ahead forecast for the model from Q7
ar0garch11.forecast <- ugarchforecast(ar0garch11.fit,n.ahead = 10)
ar0garch11.forecast

# Q9: Plot the excess returns forecast.
plot(ar0garch11.forecast, which=1)

# Q10: Plot the volatility forecast
plot(ar0garch11.forecast, which=3)

##### GARCH Exercise 4

data(m.ibmspln)

# Q2: Estimate a GARCH(1,1)-M model for the S&P500 excess returns series. Determine if the effect of volatility on asset returns is significant
sp <- as.ts(m.ibmspln)
sp <- sp[,2]
sp <- ts(sp, start = c(1926,1), end = c(1999,12), frequency = 12)

garch11m = ugarchspec(variance.model = list(garchOrder=c(1,1)),
                          mean.model = list(armaOrder=c(0,0),
                                            archm=TRUE,
                                            archpow=2))
garch11m.fit = ugarchfit(spec=garch11m,data=sp)
garch11m.fit

# Q3:Excess IBM stock returns are defined as a regular zoo variable. Convert this to a time series variable with correct dates
ibm <- as.ts(m.ibmspln)
ibm <- ibm[,1]
ibm <- ts(ibm, start = c(1926,1), frequency = 12)

# Q4: Plot the absolute and squared excess IBM stock returns along with its ACF and PACF graphs and determine the appropriate model configuration.
plot(ibm)
Acf(ibm)
Pacf(ibm)
plot(ibm^2)
Acf(ibm^2)
Pacf(ibm^2)

# Q5: The exponential GARCH model incorporates asymmetric effects for positive and negative asset returns. Estimate an AR(1)-EGARCH(1,1) model for the IBM series
ar1egarch11 <- ugarchspec(variance.model = list(model="eGARCH",
                                                variance.targeting=TRUE,
                                                garchOrder=c(1,1)),
                          mean.model = list(armaOrder=c(1,0)),
                         distribution.model = "ged")
ar1egarch11.fit = ugarchfit(spec=ar1egarch11,data=ibm)
ar1egarch11.fit

# Q6: Using the results from Q5, get rolling window forecasts starting from the 800th observation and refit the model after every three observations
rollfore <- ugarchroll(ar1egarch11, ibm, n.start = 800, refit.every = 3,
                   refit.window = "moving", solver = "hybrid",
                   calculate.VaR = TRUE, VaR.alpha = c(0.01, 0.05),
                   keep.coef = TRUE)
show(rollfore)
report(rollfore, type="VaR", VaR.alpha = 0.01, conf.level = 0.95) #backtesting

preds <- as.data.frame(rollfore) # Yo safe data as a dataframe
head(preds)
garchvol <- xts(preds$Sigma, order.by = as.Date(rownames(preds)))
plot(garchvol)

# Q6.1: Evaluate the accuracy of preds$Mu and preds$Sigma by comparing it with preds$Realized

# Prediction error for the mean
e  <- preds$Realized - preds$Mu
mean(e^2)

# Prediction error for the variance
d  <- e^2 - preds$Sigma^2
mean(d^2)

# Q6.2 How much would you lose in the best of the 5% worst cases?

garchspec.1 <- ugarchspec(mean.model = list(armaOrder = c(1,0)),
                        variance.model = list(model = "gjrGARCH"),
                        distribution.model = "sstd")


garchroll <- ugarchroll(garchspec.1, data = ibm, n.start = 800, refit.window = "moving", refit.every = 3)
```

```r
garchVaR <- quantile(as.data.frame(garchroll)$Realized, probs = 0.05)
garchVaR <- quantile(garchroll, probs = 0.05)

actual <- xts(as.data.frame(garchroll)$Realized, time(garchVaR))
VaRplot(alpha = 0.05, actual = actual, VaR = garchVaR)


# Q7: Estimate an AR(1)-GARCH(1,1) model for the IBM series and get a) forecast volatility and b) a bootstrap forecast for the next 50 periods with 500 replications. sGARCh is standar garch model !
ar1garch11 <- ugarchspec(variance.model = list(model="sGARCH",
                                               garchOrder=c(1,1)),
                          mean.model = list(armaOrder=c(1,0)),
                         distribution.model = "norm")
ar1garch11.fit <- ugarchfit(spec=ar1garch11,data=ibm)
ar1garch11.fit

# Use the method sigma to retrieve the estimated volatilities
garchvol <- sigma(ar1garch11.fit)

# Plot the volatility for 2017
plot(garchvol["2017"])

# Compute unconditional volatility
sqrt(uncvariance(garchfit))

# Compute the annualized volatility
annualvol <- sqrt(252) * sigma(garchfit)

# Forecast volatility 5 days ahead and add
garchforecast <- ugarchforecast(fitORspec = garchfit, n.ahead = 5)

# Extract the predicted volatilities and print them
print(sigma(garchforecast))


# b) bootstrap forecast
bootp <- ugarchboot(ar1garch11.fit, method = c("Partial", "Full")[2],
                   n.ahead = 50, n.bootpred = 500)
show(bootp)

# Q8: Plot the forecasted returns and sigma with bootstrap error bands.
plot(bootp, which=2)
plot(bootp, which=3)

# Q9: We can use Monte-Carlo simulation to get a distribution of the parameter estimates. Using the fitted model from Q7, run the simulation for 500 periods for a horizon of 2000 periods
dist <- ugarchdistribution(ar1garch11.fit, n.sim = 2000, n.start = 1,
                          m.sim = 500)
show(dist)

# Q10: Plot the density functions of the parameter estimates.
plot(dist, which=1)

############ GARCH Exercise 5
library(parallel)
library(rugarch)

# Q10: Estimate a GARCH model with fat tails and other stilized facts for IBM returns. The skewed student t distribution has two extra params: degree of freedom, the lower is v, the fatter the tails skewness
 parameter, <1 negative skewness, >1 positive skewness

# Specify a standard GARCH model with normal distrib
garchspec.norm <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "sGARCH"),
                 distribution.model = "norm")

# Specify a standard GARCH model with skewed student t
garchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "sGARCH"),
                 distribution.model = "sstd")

# Estimate the model
garchfit.norm <- ugarchfit(data = rets , spec = garchspec.norm)
garchfit <- ugarchfit(data = rets , spec = garchspec)

# Use the method sigma to retrieve the estimated volatilities
garchvol.norm <- sigma(garchfit.norm)
garchvol <- sigma(garchfit)

# Plot the volatility
plot(garchvol.norm)
plot(garchvol)

coef(garchfit.norm)
coef(garchfit)

# Estimated standardized returns
stdret.norm <- residuals(garchfit.norm, standardize = TRUE)
library(PerformanceAnalytics)
chart.Histogram(stdret.norm, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))

# Estimated standardized returns with t-studnet
stdret <- residuals(garchfit, standardize = TRUE)
chart.Histogram(stdret, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))


# Specify a gjrGARCH model with skewed student t
garchspec.gjr <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "gjrGARCH"),
                 distribution.model = "sstd")

# Estimate the model
garchfit.gjr <- ugarchfit(data = rets, spec = garchspec)

# Use the method sigma to retrieve the estimated volatilities
garchvol.gjr <- sigma(garchfit.gjr)

# Plot the volatility
plot(garchvol.gjr)

coef(garchfit.gjr)

# Estimated standardized returns
stdret.gjr <- residuals(garchfit.gjr, standardize = TRUE)
chart.Histogram(stdret.gjr, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))

# Visualize volatility response using newsimpact()
out <- newsimpact(garchfit)
plot(out$zx, out$zy, xlab="prediction error", ylab="predicted variance")

out.gjr <- newsimpact(garchfit.gjr)
plot(out.gjr$zx, out.gjr$zy, xlab="prediction error", ylab="predicted variance")

# How much would you lose in the best of the 5% worst cases?
# Using skewed student t
garchroll <- ugarchroll(garchspec, data = rets, n.start = 252, refit.window = "moving", refit.every = 90)
garchVaR <- quantile(as.data.frame(garchroll)$Realized, probs = 0.05)
```

```
garchVaR <- quantile(garchroll, probs = 0.05)

actual <- xts(as.data.frame(garchroll)$Realized, time(garchVaR))
VaRplot(alpha = 0.05, actual = actual, VaR = garchVaR)

# Calculation of coverage for S&P 500 returns and 5% probability level
mean(actual < garchVaR)

# interpretation: Valid prediction model has a coverage that is close to the probability level α used. If coverage » α: too many exceedances: the predicted quantile should be more negative. Risk of losing
 money has been underestimated. If coverage « α: too few exceedances, the predicted quantile was too negative. Risk of losing money has been overestimated.
```

```
############ Exercise 6 (https://rpubs.com/yevonnael/garch-models-demo)

library(tidyquant)
GOOG <- getSymbols(Symbols = "KO", from = "2000-01-01", to = "2018-01-01", src = "yahoo", adjust=TRUE, auto.assign = FALSE)

# Take the adjusted price only
GOOG <- Ad(GOOG)
# Plot daily stock price
plot(GOOG)
# Compute daily returns
GOOG_ret <- CalculateReturns(GOOG) %>% na.omit()
# Plot daily returns
plot(GOOG_ret)
# Compute the annualized volatility for the complete sample
sqrt(252) * sd(GOOG_ret)
# Compute the annualized standard deviation for the year 2009
sqrt(252) * sd(GOOG_ret["2009"])
# Compute the annualized standard deviation for the year 2017
sqrt(252) * sd(GOOG_ret["2017"])
# Load the package PerformanceAnalytics
library(PerformanceAnalytics)

# Showing two plots on the same figure
par(mfrow=c(2,1))

# Compute the rolling 1 month estimate of annualized volatility
chart.RollingPerformance(R = GOOG_ret, width = 22,
     FUN = "sd.annualized", scale = 252, main = "One month rolling volatility")

# Compute the rolling 3 months estimate of annualized volatility
chart.RollingPerformance(R = GOOG_ret, width = 66,
     FUN = "sd.annualized", scale = 252, main = "Three months rolling volatility")

# The GARCH equation for volatility prediction

library(rugarch)

# Specify a standard GARCH model with constant mean
garchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "sGARCH"),
                 distribution.model = "norm")

# Estimate the model
garchfit <- ugarchfit(data = GOOG_ret, spec = garchspec)

# Use the method sigma to retrieve the estimated volatilities
garchvol <- sigma(garchfit)

# Plot the volatility for 2017
plot(garchvol["2017"])

# Compute unconditional volatility
sqrt(uncvariance(garchfit))

# Print last 10 ones in garchvol
tail(garchvol, 10)

# Forecast volatility 5 days ahead and add
garchforecast <- ugarchforecast(fitORspec = garchfit,
                     n.ahead = 5)

# Extract the predicted volatilities and print them
print(sigma(garchforecast))

# Compute the annualized volatility
annualvol <- sqrt(252) * sigma(garchfit)

# Compute the 5% vol target weights
vt_weights <- 0.05 / annualvol

# Compare the annualized volatility to the portfolio weights in a plot
plot(merge(annualvol, vt_weights), multi.panel = TRUE)

# Non-normality of standardized returns

# Estimated standardized returns
stdret <- residuals(garchfit, standardize = TRUE)
library(PerformanceAnalytics)
chart.Histogram(GOOG_ret, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))

# Specify a standard GARCH model with skewed student t
garchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "sGARCH"),
                 distribution.model = "sstd")

# Estimate the model
garchfit <- ugarchfit(data = GOOG_ret, spec = garchspec)

# Use the method sigma to retrieve the estimated volatilities
garchvol <- sigma(garchfit)

# Plot the volatility for 2017
plot(garchvol["2017"])

coef(garchfit)

# Estimated standardized returns
stdret <- residuals(garchfit, standardize = TRUE)
library(PerformanceAnalytics)
chart.Histogram(stdret, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))

# Leverage effect
```

```
# Specify a gjrGARCH model with skewed student t
garchspec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                 variance.model = list(model = "gjrGARCH"),
                 distribution.model = "sstd")

# Estimate the model
garchfit <- ugarchfit(data = GOOG_ret, spec = garchspec)

# Use the method sigma to retrieve the estimated volatilities
garchvol <- sigma(garchfit)

# Plot the volatility for 2017
plot(garchvol["2017"])



coef(garchfit)
# Estimated standardized returns
stdret <- residuals(garchfit, standardize = TRUE)
library(PerformanceAnalytics)
chart.Histogram(stdret, methods = c("add.normal", "add.density"),
          colorset=c("gray","red","blue"))

out <- newsimpact(garchfit)
plot(out$zx, out$zy, xlab="prediction error", ylab="predicted variance")
```