```
# load required packages
library(quantmod)
library(forecast)
library(lmtest)
library(xts)


# Download datasets
startDate <- "2018-02-01"
endDate <- "2018-12-30"

getSymbols("JPM", src="yahoo", from=startDate, to=endDate )
JPMorgan <- JPM[, "JPM.Adjusted", drop = FALSE]

getSymbols("^GSPC", src="yahoo", from=startDate, to=endDate)
SP500 <- GSPC[, "GSPC.Adjusted", drop=FALSE]

allData <- data.frame(JPMorgan, SP500)


# Explore the dataset
head(allData)
str(allData)
summary(allData)

# 1.1 Calculate Average stock value
average_value <- mean(JPMorgan)
average_value
# [1] 107.2015

# 1.2 Calculate Stock volatility
stock_volatility <- sd(JPMorgan)
stock_volatility
# [1] 4.56665


# 1.3 Calculate Daily stock returns
# 1.3.1 Daily simple returns
simple_returns <- diff(JPMorgan)/lag(JPMorgan)[-1]
names(simple_returns) <- "JPM.simpleReturns"
head(simple_returns)

#             JPM.simpleReturns
# 2018-02-02      -0.022161369
# 2018-02-05      -0.047952357
# 2018-02-06       0.030422834
# 2018-02-07       0.006779018
# 2018-02-08      -0.044210207
# 2018-02-09       0.020022238

# 1.3.1.1 calculate daily and annualized volatility
simpleReturns_volatility <- sd(simple_returns)
simpleReturns_volatility
# [1] 0.01438354
```

```
annualized_volatilitySimple <- simpleReturns_volatility * sqrt(252)
annualized_volatilitySimple
# [1] 0.2283317


# 1.3.2 Daily continously compounded returns

comp_returns <- diff(log(JPMorgan))[-1]
names(comp_returns) <- "JPM.compReturns"
head(comp_returns)
#            JPM.compReturns
# 2018-02-02    -0.022410622
# 2018-02-05    -0.049140201
# 2018-02-06     0.029969236
# 2018-02-07     0.006756144
# 2018-02-08    -0.045217271
# 2018-02-09     0.019824429

# 1.3.2.1 calculate daily and annualized volatility
compReturns_volatility <- sd(comp_returns)
compReturns_volatility
# [1] 0.01441866

annualized_volatilityComp <- compReturns_volatility * sqrt(252)
annualized_volatilityComp
# [1] 0.2288891




# 3.1.2 Linear regression
# Implement a two variable regression
linear_model <- lm(JPM.Adjusted  ~ GSPC.Adjusted, data=allData)
summary(linear_model)
#
# Call:
#   lm(formula = JPM.Adjusted ~ GSPC.Adjusted, data = allData)
#
# Residuals:
#   Min      1Q  Median      3Q     Max
# -6.7551 -2.3973  0.4835  2.3838  5.6483
#
# Coefficients:
#                Estimate  Std. Error t value  Pr(>|t|)
# (Intercept)   13.751225   5.294731   2.597     0.01 *
# GSPC.Adjusted  0.034065   0.001929  17.662   <2e-16 ***
#   ---
#   Signif. codes:  0 â€˜***â€™ 0.001 â€˜**â€™ 0.01 â€˜*â€™ 0.05 â€˜.â€™
0.1 â€˜ â€™ 1

# Residual standard error: 2.97 on 227 degrees of freedom
# Multiple R-squared:  0.5788,    Adjusted R-squared:  0.5769
# F-statistic: 311.9 on 1 and 227 DF,  p-value: < 2.2e-16
```

```
# 3.1.3 Univariate Time Series Analysis
# Forecast S&P/Case-Shiller U.S National Home Price Index using ARMA
model
startdate <- "1978-01-01"
getSymbols("CSUSHPINSA", src='FRED', from = "1978-01-01" )

# Exploratory data analysis
head(CSUSHPINSA)
str(CSUSHPINSA)
attr(CSUSHPINSA, "dimnames")
class(CSUSHPINSA)
summary(CSUSHPINSA)


# 3.1.3.1 Implement Augmented Dickey-Fuller test
adf.test(CSUSHPINSA)
# Augmented Dickey-Fuller Test
#
# data:  CSUSHPINSA
# Dickey-Fuller = -2.3243, Lag order = 7, p-value = 0.4402
# alternative hypothesis: stationary


# 3.1.3.2 Implement ARIMA(p,d,q)
# plot acf and pacf
plot(CSUSHPINSA)
acf(CSUSHPINSA)
pacf(CSUSHPINSA)


plot(log(CSUSHPINSA))
acf(log(CSUSHPINSA))
pacf(log(CSUSHPINSA))

plot(diff(CSUSHPINSA))
acf(diff(CSUSHPINSA))
pacf(diff(CSUSHPINSA))



plot(diff(diff(CSUSHPINSA)))
acf(diff(diff(CSUSHPINSA)))
pacf(diff(diff(CSUSHPINSA)))


# implement ARIMA model
```

```
auto.arima(CSUSHPINSA)
# Series: CSUSHPINSA
# ARIMA(3,1,2) with drift
#
# Coefficients:
#   ar1     ar2      ar3     ma1     ma2   drift
# 0.8592  0.1036  -0.2281  0.6294  0.2962  0.3785
# s.e.  0.1446  0.2154   0.1135  0.1412  0.0789  0.1134
#
# sigma^2 estimated as 0.09747:  log likelihood=-97.53
# AIC=209.07   AICc=209.36   BIC=236.81

arima_model <- arima(CSUSHPINSA, c(3,2,1))
coeftest(arima_model)
summary(arima_model)

plot(forecast(arima_model), include = 20)


# Run the following commands to plot the series and fitted values
# ? overfitting here
ts.plot(CSUSHPINSA)
model_fitted <- CSUSHPINSA - residuals(arima_model)
points(model_fitted, type = "l", col = 7, lty = 2)


# using in-sample forecast
len <- length(CSUSHPINSA)
updiv <- 0.9 * len
trainset <- CSUSHPINSA[1:updiv]
testset <- CSUSHPINSA[(updiv + 1):len]
lentest <- length(testset)


# fit non-seasonal model
arima_model2 <- arima(trainset, c(3,2,1))
preds <- predict(arima_model2, n.ahead=lentest)$pred
ar_forecast <- forecast(arima_model2, h=25)
plot(ar_forecast)
accuracy(preds, testset)[2]
# [1] 2.526616 RMSE


# fit the model with seasonality
arima_model3 <- arima(trainset, c(3,2,1),seasonal = list(order =
c(1,0,0), period = 12))
preds <- predict(arima_model3, n.ahead=lentest)$pred
ar_forecast <- forecast(arima_model3, h=25)
plot(ar_forecast)
accuracy(preds, testset)[2]
# [1] 2.179081 RMSE

# plot the predictions and testset
predxts <- as.xts(preds)
```

```
index(predxts) <- index(testset)
plot(predxts)
points(testset)


# plot in-sample predictions
plot(CSUSHPINSA)
points(predxts, col=2)


# put all in a function
arima_test <-function(x,p,d,q,P=0,D=0,Q=0,S=0){
  lendata <- length(x);
  lentrain <- 0.9 * lendata;
  train <- x[1:lentrain];
  test <- x[(lentrain + 1):lendata];
  lentestx <- length(test)

  aR <- arima(train, c(p,d,q),seasonal = list(order = c(P,D,Q), period =
S))
  predictions <- predict(aR, n.ahead=lentestx)$pred;
  aR_fc <- forecast(aR, h=25);
  accuracy(predictions, test)[2];

  predictxts <- as.xts(predictions)
  index(predictxts) <- index(test)
  plot(x)
  points(predictxts, col = 2, lty = 2)

}

arima_test(CSUSHPINSA,3,2,1,1,0,0,12)
```