FA542 - Homework #4

I pledge my honor that I have abided by the Stevens Honor System.

Sid Bhatia

2023-11-16

Problem 1

Download daily price data for January 1, 1990 through November 1, 2023 of Microsoft stock from Yahoo Finance. You may use the quantmod package in R for this purpose.

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## ###################################### WARNING ######################################
## # We noticed you have dplyr installed. The dplyr lag() function breaks how    #
## # base R's lag() function is supposed to work, which breaks lag(my_xts).      #
## #                                                                              #
## # If you call library(dplyr) later in this session, then calls to lag(my_xts) #
## # that you enter or source() into this session won't work correctly.           #
## #                                                                              #
## # All package code is unaffected because it is protected by the R namespace   #
## # mechanism.                                                                   #
## #                                                                              #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## # You can use stats::lag() to make sure you're not using dplyr::lag(), or you #
## # can add conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop    #
## # dplyr from breaking base R's lag() function.                                 #
## ###################################### WARNING ######################################
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# Set the date range.
start_date <- as.Date("1990-01-01")
end_date <- as.Date("2023-11-01")
```

```r
# Download the data.
getSymbols("MSFT", src = "yahoo", from = start_date, to = end_date)
```

```
## [1] "MSFT"
```

```r
# View the data.
head(MSFT)
```

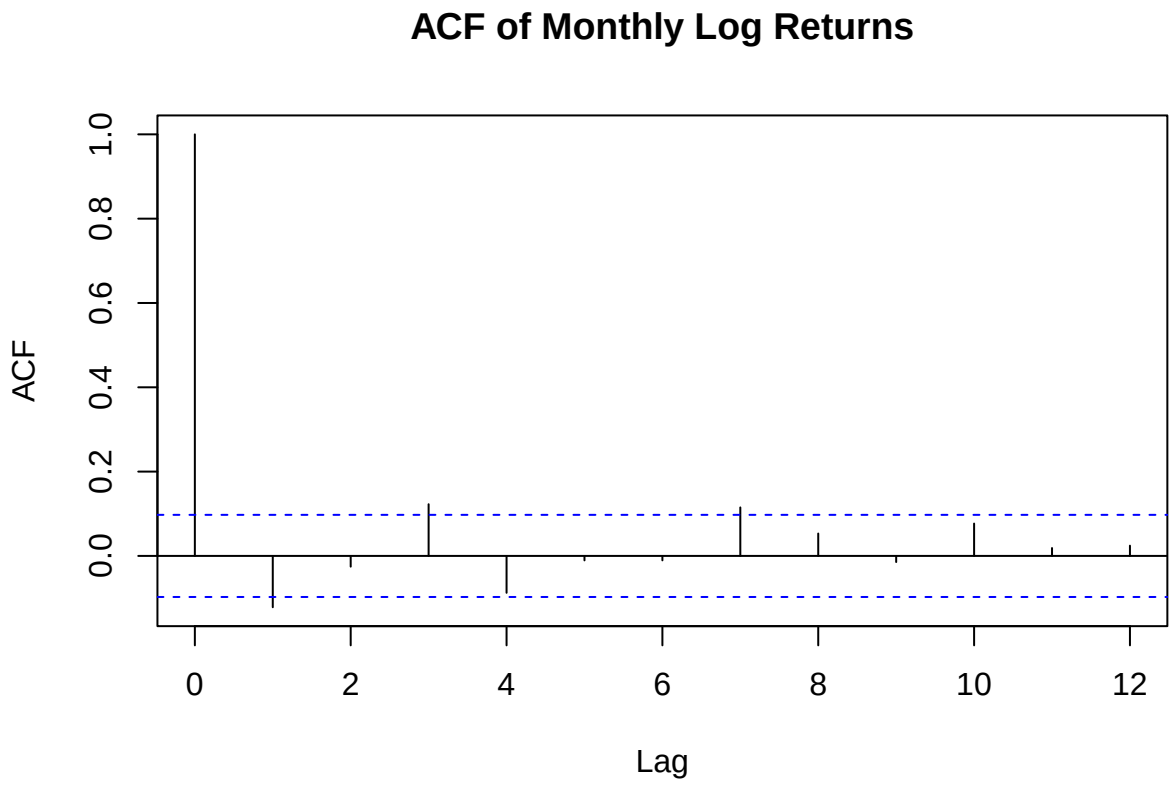
```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 1990-01-02  0.605903  0.616319 0.598090   0.616319    53035200     0.3820956
## 1990-01-03  0.621528  0.626736 0.614583   0.619792   113774400     0.3842487
## 1990-01-04  0.619792  0.638889 0.616319   0.638021   125740800     0.3955499
## 1990-01-05  0.635417  0.638889 0.621528   0.622396    69566400     0.3858630
## 1990-01-08  0.621528  0.631944 0.614583   0.631944    58982400     0.3917824
## 1990-01-09  0.631944  0.638889 0.626736   0.630208    70300800     0.3907062
```

a. Is there any evidence of serial correlations in the monthly log returns. Use autocorrelations and 5% significance level to answer the question. If yes, remove the serial correlations.

```r
library(forecast)

# Calculate monthly returns.
monthly_prices <- to.monthly(MSFT, indexAt='lastof', OHLC=FALSE)
monthly_log_returns <- diff(log(Cl(monthly_prices)), lag=1)

# Remove the first NA value.
monthly_log_returns <- na.omit(monthly_log_returns)

head(monthly_log_returns)
```

```
##             MSFT.Close
## 1990-02-28  0.06538314
## 1990-03-31  0.11468349
## 1990-04-30  0.04631566
## 1990-05-31  0.23001595
## 1990-06-30  0.04027431
## 1990-07-31 -0.13353201
```

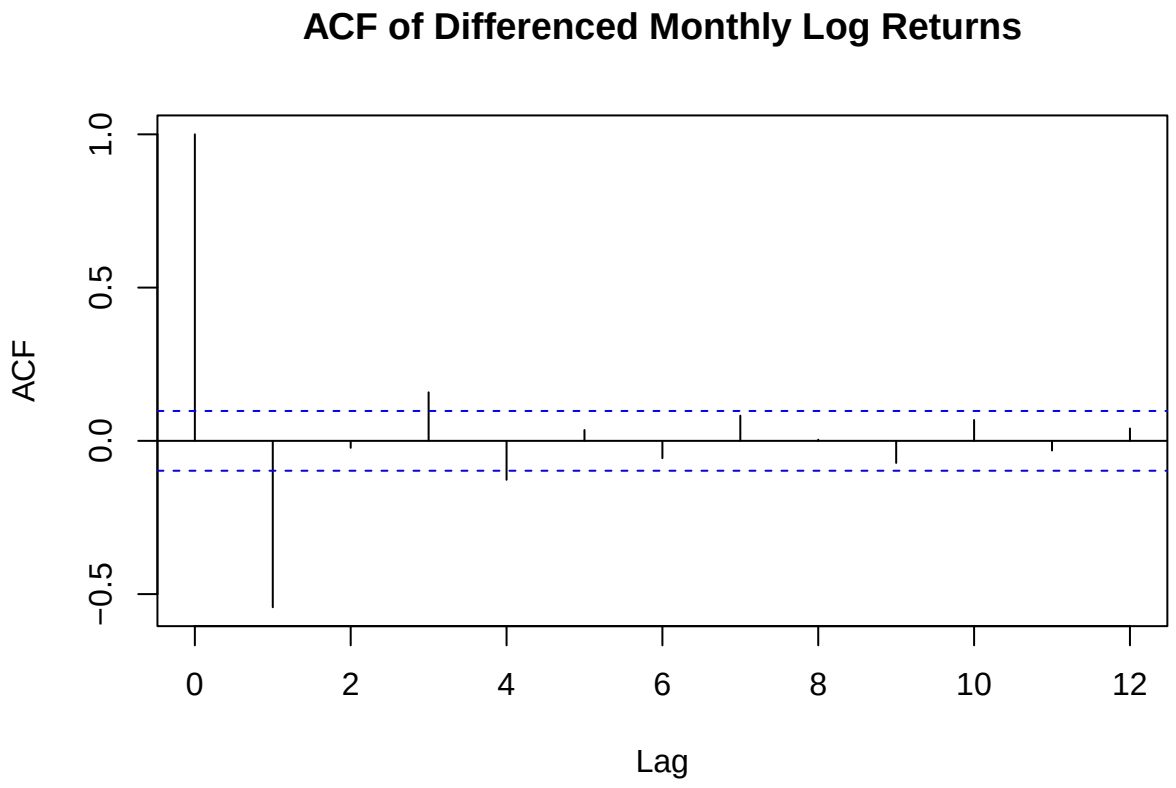
```r
# Check for serial correlation using ACF.
acf(monthly_log_returns, lag.max=12, main="ACF of Monthly Log Returns")
```




```
# Perform a statistical test for autocorrelation.
Box.test(monthly_log_returns, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  monthly_log_returns
## X-squared = 6.0476, df = 1, p-value = 0.01393
```

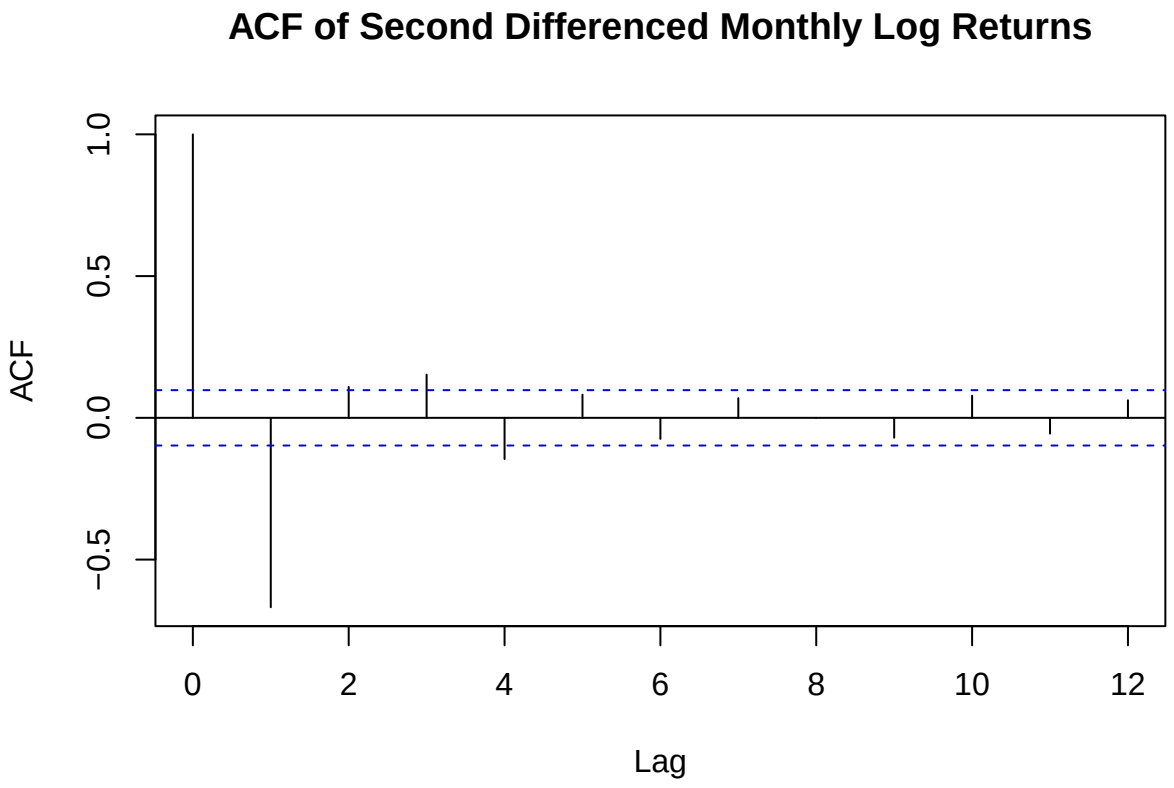
```
# Calculate the first difference of the monthly log returns.
diff_log_returns <- diff(monthly_log_returns)

# Removing the NA value that gets introduced by differencing.
diff_log_returns <- na.omit(diff_log_returns)

# Check again for serial correlation using ACF.
acf(diff_log_returns, lag.max=12, main="ACF of Differenced Monthly Log Returns")
```




```
# Perform the Box test again.
Box.test(diff_log_returns, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  diff_log_returns
## X-squared = 119.95, df = 1, p-value < 2.2e-16
```

```
# Calculate the second difference of the monthly log returns.
second_diff_log_returns <- diff(diff_log_returns)

# Removing the NA value that gets introduced by differencing.
second_diff_log_returns <- na.omit(second_diff_log_returns)

# Check again for serial correlation using ACF.
acf(second_diff_log_returns, lag.max=12, main="ACF of Second Differenced Monthly Log Returns")
```




```
# Perform the Box test again.
Box.test(second_diff_log_returns, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  second_diff_log_returns
## X-squared = 181.23, df = 1, p-value < 2.2e-16
```

```
auto_arima_model <- auto.arima(monthly_log_returns)

# Check the residuals of the ARIMA model.
arima_residuals <- residuals(auto_arima_model)

# Perform the Box test on the residuals.
Box.test(arima_residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  arima_residuals
## X-squared = 0.0058547, df = 1, p-value = 0.939
```

Since p-value for ARIMA model residuals is greater than 0.05 for the Ljung-Box test, we can reject the null hypothesis that there are serial autocorrelations, implying we have removed them with this ARIMA($p = 1, d = 0, q = 2$) model.

b. Is there any evidence of ARCH effects in the monthly log returns? Use the residual series if there are serial correlations in part (ii). Use Ljung-Box statistics for the squared returns (or residuals) with 6 and 12 lags of autocorrelations and 5% significance level to answer the question.

```
# Calculate squared returns.
squared_returns <- monthly_log_returns^2

# Ljung-Box test on squared returns for 6 and 12 lags.
Box.test(squared_returns, lag = 6, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  squared_returns
## X-squared = 66.491, df = 6, p-value = 2.14e-12
```

```
Box.test(squared_returns, lag = 12, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  squared_returns
## X-squared = 140.5, df = 12, p-value < 2.2e-16
```

```
# Calculated squared residuals.
squared_residuals <- arima_residuals^2

# Ljung-Box test on squared returns for 6 and 12 lags.
Box.test(squared_residuals, lag = 6, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  squared_residuals
## X-squared = 70.746, df = 6, p-value = 2.874e-13
```

```
Box.test(squared_residuals, lag = 12, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  squared_residuals
## X-squared = 157, df = 12, p-value < 2.2e-16
```

After conducting Ljung-Box tests for both squared returns and the squared residuals of the ARIMA$(1, 0, 2)$ model, both tests fail with 6 and 12 lags at the 5% significance level. As such, there is evidence of ARCH effects in the monthly log returns.

c. Identify an ARCH model for the data and fit the identified model. Write down the fitted model and justify your choice of parameters.

```
# Load the required library
library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
# Specify the GARCH(1,1) model
spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                   mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                   distribution.model = "std")

# Fit the model
fit <- ugarchfit(spec = spec, data = monthly_log_returns)

fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## omega   0.000255    0.000134   1.8957 0.058004
## alpha1  0.092012    0.030501   3.0167 0.002555
## beta1   0.870489    0.036997  23.5285 0.000000
## shape   9.911135    4.211369   2.3534 0.018601
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega   0.000255    0.000107   2.3905 0.016826
## alpha1  0.092012    0.024933   3.6903 0.000224
## beta1   0.870489    0.026119  33.3282 0.000000
## shape   9.911135    3.559197   2.7847 0.005358
##
## LogLikelihood : 451.939
##
## Information Criteria
## ------------------------------------
##
## Akaike       -2.2120
## Bayes        -2.1725
## Shibata      -2.2122
## Hannan-Quinn -2.1964
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.501  0.1138
## Lag[2*(p+q)+(p+q)-1][2]     2.573  0.1832
```

```
## Lag[4*(p+q)+(p+q)-1][5]     4.159  0.2349
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       1.259  0.2618
## Lag[2*(p+q)+(p+q)-1][5]     3.047  0.3986
## Lag[4*(p+q)+(p+q)-1][9]     4.253  0.5466
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     2.081 0.500 2.000  0.1492
## ARCH Lag[5]     3.221 1.440 1.667  0.2593
## ARCH Lag[7]     3.700 2.315 1.543  0.3920
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.6486
## Individual Statistics:
## omega  0.15771
## alpha1 0.31389
## beta1  0.28325
## shape  0.06685
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.2335 0.8155
## Negative Sign Bias  0.2552 0.7987
## Positive Sign Bias  0.5252 0.5997
## Joint Effect        1.2447 0.7423
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     39.59    0.0036997
## 2    30     54.33    0.0029613
## 3    40     78.85    0.0001645
## 4    50     72.65    0.0157157
##
##
## Elapsed time : 0.05084085
```

As such, the GARCH$(1, 1)$ model is as follows:

$$\sigma_t^2 = 0.000255 + 0.092012e_{t-1}^2 + 0.870489\sigma_{t-1}^2$$

Problem 2

Use the following commands to simulate a two-regime TAR(1) model with 400 observations

require(NTS) set.seed(1) phi <- matrix(c(0.8,-0.7),2,1) m1 <- uTAR.sim(400, c(1,1), phi) xt <- m1$series

a. Obtain a time plot for the data and its sample autocorrelation function with 12 lags.

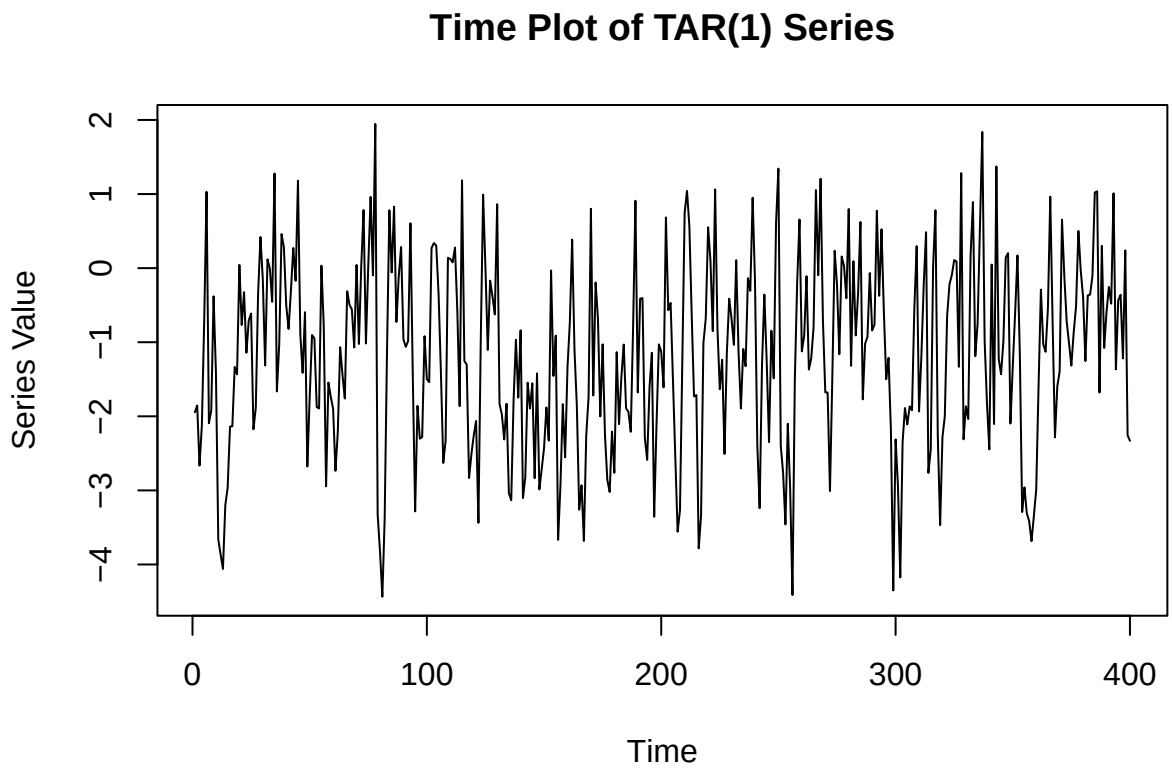
```
require(NTS)
```

```
## Loading required package: NTS
```

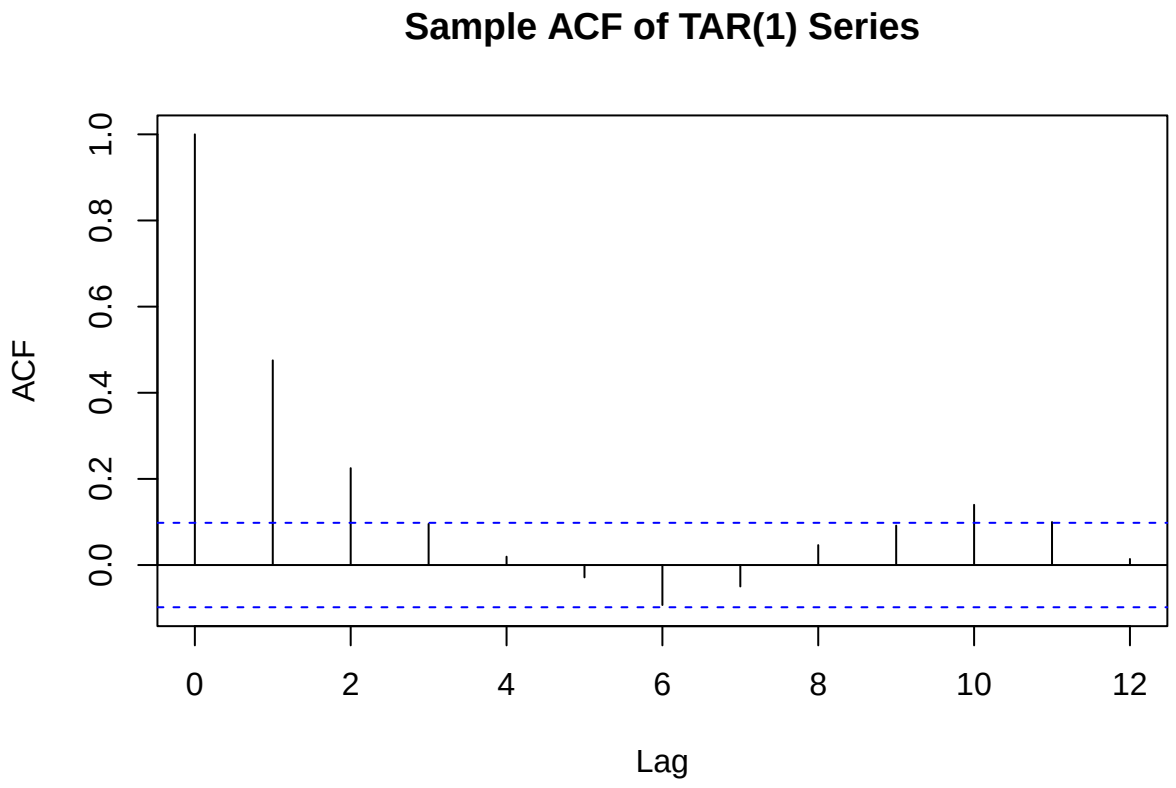
```
set.seed(1)
phi <- matrix(c(0.8,-0.7),2,1)
m1 <- uTAR.sim(400, c(1,1), phi)
xt <- m1$series

head(xt)
```

```
## [1] -1.9459316 -1.8536140 -2.6661334 -2.1216140 -0.7056902  1.0294153
```

```
# Time plot
plot(xt, type = "l", main = "Time Plot of TAR(1) Series", xlab = "Time", ylab = "Series Value")
```

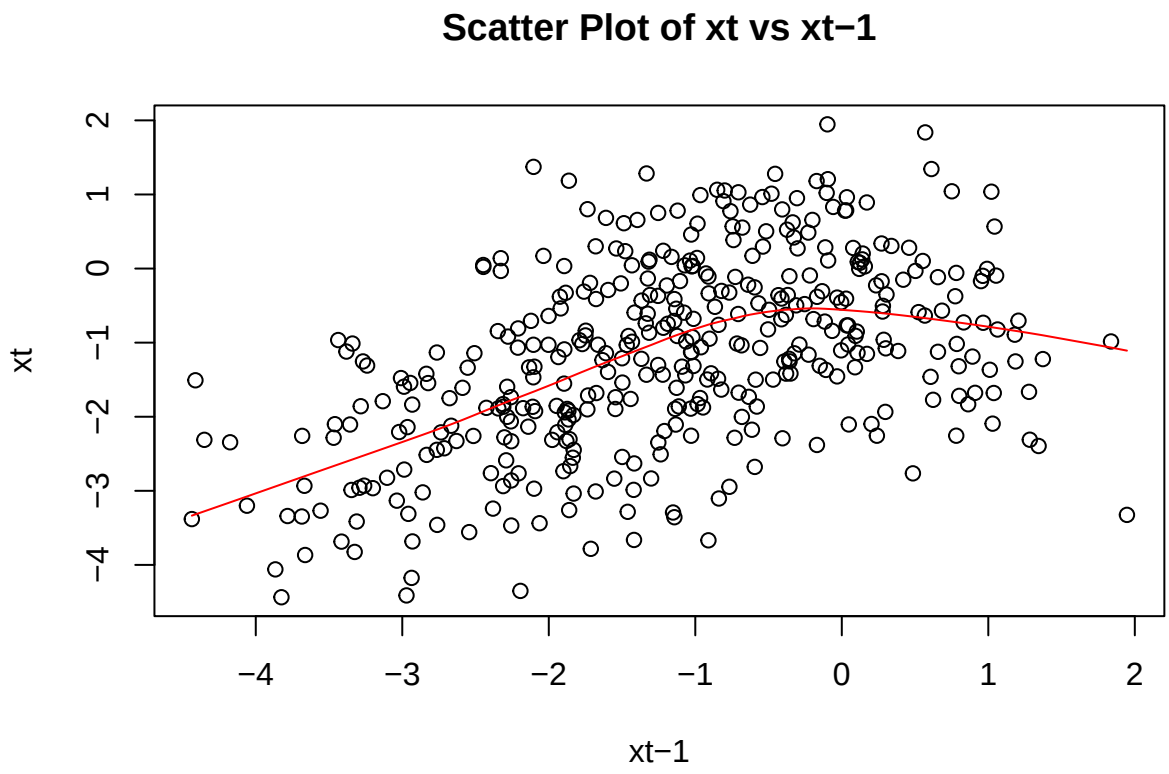



```
# Sample Autocorrelation Function (ACF) with 12 lags
acf(xt, lag.max = 12, main = "Sample ACF of TAR(1) Series")
```



b. Obtain a scatter plot for x_t versus x_{t-1} and draw a smooth line on the plot using loess local smoothing.

```r
# Scatter plot of xt vs xt-1 with loess smoothing.
xt_lag <- xt[-length(xt)]
xt_current <- xt[-1]
plot(xt_lag, xt_current, main = "Scatter Plot of xt vs xt-1", xlab = "xt-1", ylab = "xt")
lines(lowess(xt_lag, xt_current), col = "red")
```



c. Apply threshold tests to confirm the threshold nonlinearity of the time series.

```
library(tseries)
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
# ADF test
adf.test <- adf.test(xt)
```

```
## Warning in adf.test(xt): p-value smaller than printed p-value
```

```
# KPSS test
kpss.test <- kpss.test(xt)
```

```
## Warning in kpss.test(xt): p-value greater than printed p-value
```

```
# Perform Tsay's test for threshold nonlinearity.
tsay.test <- Tsay.test(xt)

# Display the results.
adf.test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  xt
## Dickey-Fuller = -6.1491, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  xt
## KPSS Level = 0.22774, Truncation lag parameter = 5, p-value = 0.1
```

```
tsay.test
```

```
## $test.stat
## [1] 32.05
##
## $p.value
## [1] 2.884e-08
##
## $order
## [1] 1
```

d. Build a two-regime TAR model for the series and write down the fitted model.

```
library(TSA)

# Fit a two-regime TAR model.
tar.model <- tar(y = xt, p1 = 1, p2 = 1, d = 1)

# Summary of the model.
summary(tar.model)
```

```
##              Length Class  Mode
## dxy1         798    -none- numeric
## dxy2         798    -none- numeric
## p1             1    -none- numeric
## q1             1    -none- numeric
## d              1    -none- numeric
## qr1            4    -none- list
## qr2            4    -none- list
## x.regime1    578    -none- numeric
## y.regime1    289    -none- numeric
## x.regime2    220    -none- numeric
## y.regime2    110    -none- numeric
## thd            1    -none- numeric
```

```
## thdindex       1    -none- numeric
## qr1            4    -none- list
## qr2            4    -none- list
## i1             1    -none- numeric
## i2             1    -none- numeric
## x            399    -none- numeric
## m              1    -none- numeric
## rss1           1    -none- numeric
## rss2           1    -none- numeric
## n1             1    -none- numeric
## n2             1    -none- numeric
## std.res      399    -none- numeric
## p1             1    -none- numeric
## p2             1    -none- numeric
## rms1           1    -none- numeric
## rms2           1    -none- numeric
## is.constant1   1    -none- logical
## is.constant2   1    -none- logical
## residuals    399    -none- numeric
## AIC            1    -none- numeric
## aic.no.thd     1    -none- numeric
## y            400    -none- numeric
## like           1    -none- numeric
## method         1    -none- character
```

```
# Extracting coefficients for each regime from the TAR model.
coefficients_regime1 <- coef(tar.model$qr1)
coefficients_regime2 <- coef(tar.model$qr2)

# Extracting the threshold value.
threshold_value <- tar.model$thd

coefficients_regime1
```

```
## intercept-xt      lag1-xt
##   -0.1566838    0.7106902
```

```
coefficients_regime2
```

```
## intercept-xt      lag1-xt
##   -0.2808983   -0.7012721
```

```
threshold_value
```

```
##
## -0.3503485
```

As such, here are the models for the following regimes:

Regime 1 for $x_{t-1} \leq -0.3503485$:

$$x_t = -0.1566838 + 0.7106902x_{t-1} + a_t$$

Regime 2 for $x_{t-1} > -0.3503485$:

$$x_t = -0.2808983 - 0.7012721x_{t-1} + a_t$$

Problem 3.

Consider the monthly mean duration unemployment in US. The data is seasonally adjusted from January 1948 to August 2017 and it is available in the file Unempduration.csv. The mean duration shows an upward trend so let x_t be the first difference of the original data.

a. Build a linear AR model for x_t. Write down the fitted model and perform model checking.

```
# Load the data.
unemp_data <- read.csv("C:/Users/sbhatia2/My Drive/University/Academics/Semester V/FA542 - Time Series
```

```
head(unemp_data)
```

```
##       DATE UEMPMEAN
## 1 1/1/1948      8.9
## 2 2/1/1948      8.4
## 3 3/1/1948      8.7
## 4 4/1/1948      8.5
## 5 5/1/1948      9.1
## 6 6/1/1948      8.8
```

```
adf.test(unemp_data$UEMPMEAN)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  unemp_data$UEMPMEAN
## Dickey-Fuller = -3.0433, Lag order = 9, p-value = 0.1367
## alternative hypothesis: stationary
```

```
# First difference of the data to address the trend and stationarity.
unemp_diff <- diff(unemp_data$UEMPMEAN)

head(unemp_diff)
```

```
## [1] -0.5  0.3 -0.2  0.6 -0.3 -0.2
```

```
adf.test(unemp_diff)
```

```
## Warning in adf.test(unemp_diff): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  unemp_diff
## Dickey-Fuller = -6.2846, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Fit an AR model for differenced, stationary data.
ar_model <- ar(unemp_diff)

# Display AR model.
ar_model
```

```
##
## Call:
## ar(x = unemp_diff)
##
## Coefficients:
```

```
##       1       2       3       4       5       6       7       8
## -0.1308  0.1053  0.1464  0.0847  0.0531  0.0848  0.0639 -0.0265
##       9      10      11      12      13
##  0.0741  0.0656  0.0540 -0.0810 -0.0625
##
## Order selected 13  sigma^2 estimated as  0.3191
```

```
# Check residuals for model with Ljung-Box test.
Box.test(residuals(ar_model), type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(ar_model)
## X-squared = 0.08175, df = 1, p-value = 0.7749
```

As such, an AR(13) model fits best for the differenced data:

$$x_t = -0.1308x_{t-1}+0.1053x_{t-2}+0.1464x_{t-3}+0.0847x_{t-4}+0.0531x_{t-5}+0.0848x_{t-6}+0.0639x_{t-7}-0.0265x_{t-8}+0.0741x_{t-9}+0.06$$

b. Perform threshold tests to confirm that x_t exhibits threshold nonlinearity.

```
kpss.test <- kpss.test(unemp_diff)
```

```
## Warning in kpss.test(unemp_diff): p-value greater than printed p-value
```

```
# Perform Tsay's test for threshold nonlinearity.
tsay.test <- Tsay.test(unemp_diff)

# Display the test results.
kpss.test
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  unemp_diff
## KPSS Level = 0.066879, Truncation lag parameter = 6, p-value = 0.1
```

```
tsay.test
```

```
## $test.stat
## [1] 1.628
##
## $p.value
## [1] 0.000426
##
## $order
## [1] 13
```

c. Use $p = 5$ and $d \in 1, \cdots, 5$, where d denotes the delay for selection of the threshold variable x_{t-d}. You may use the p value of the threshold test to select d.

```
optimal_d <- NULL
min_p_value <- 1

# Testing different delays d
for (d in 1:5) {
```

```
    tsay_test <- Tsay.test(unemp_diff, lag = 5, d = d)
    if (tsay_test$p.value < min_p_value) {
        min_p_value <- tsay_test$p.value
        optimal_d <- d
    }
}

# optimal_d now contains the delay with the smallest p-value
print(optimal_d)
```

```
## [1] 1
```

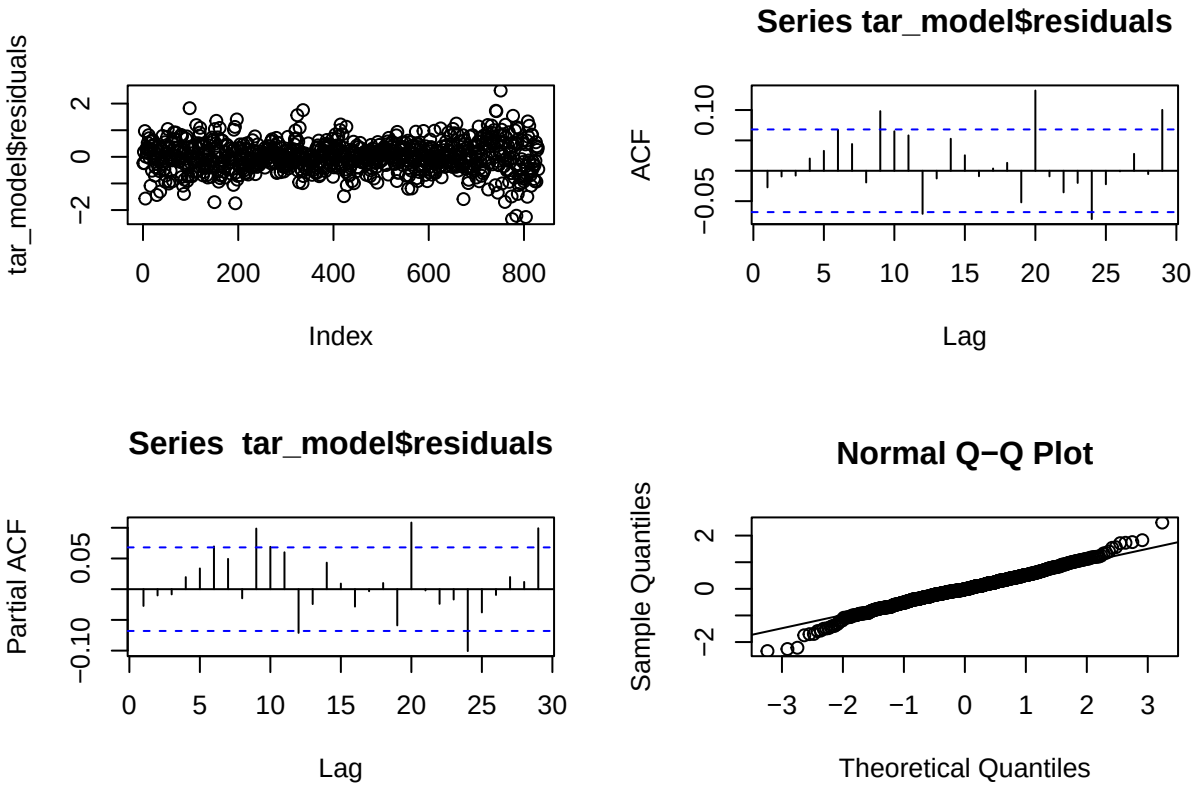
d. Build a two-regime TAR(5) model for x_t. Perform model checking and write down the fitted model.

```
# Building a two-regime TAR model
tar_model <- tar(y = unemp_diff, p1 = 5, p2 = 5, d = optimal_d)

# Model summary
summary(tar_model)
```

```
##              Length Class  Mode
## dxy1         1660   -none- numeric
## dxy2         4150   -none- numeric
## p1              1   -none- numeric
## q1              1   -none- numeric
## d               1   -none- numeric
## qr1             4   -none- list
## qr2             4   -none- list
## x.regime1     880   -none- numeric
## y.regime1     440   -none- numeric
## x.regime2    1950   -none- numeric
## y.regime2     390   -none- numeric
## thd             1   -none- numeric
## thdindex        1   -none- numeric
## qr1             4   -none- list
## qr2             4   -none- list
## i1              1   -none- numeric
## i2              1   -none- numeric
## x             830   -none- numeric
## m               1   -none- numeric
## rss1            1   -none- numeric
## rss2            1   -none- numeric
## n1              1   -none- numeric
## n2              1   -none- numeric
## std.res       830   -none- numeric
## p1              1   -none- numeric
## p2              1   -none- numeric
## rms1            1   -none- numeric
## rms2            1   -none- numeric
## is.constant1    1   -none- logical
## is.constant2    1   -none- logical
## residuals     830   -none- numeric
## AIC             1   -none- numeric
## aic.no.thd      1   -none- numeric
```

```
## y             835   -none- numeric
## like            1   -none- numeric
## method          1   -none- character
```

```
# Model checking with diagnostic plots
par(mfrow=c(2,2))
plot(tar_model$residuals)
acf(tar_model$residuals)
pacf(tar_model$residuals)
qqnorm(tar_model$residuals)
qqline(tar_model$residuals)
```




```
# Extracting coefficients and threshold value
coefficients_regime1 <- coef(tar_model$qr1)
coefficients_regime2 <- coef(tar_model$qr2)
threshold_value <- tar_model$thd

# Printing the coefficients and threshold
print(coefficients_regime1)
```

```
## intercept-unemp_diff      lag1-unemp_diff
##          -0.02339012          -0.18709330
```

```
print(coefficients_regime2)
```

```
## intercept-unemp_diff      lag1-unemp_diff      lag2-unemp_diff
##         -0.005189495         -0.086354193          0.209213849
##      lag3-unemp_diff      lag4-unemp_diff
```

```
##           0.242253421           0.131535088
```

```
print(threshold_value)
```

```
##
## 0
```

As such, here are the models for the following regimes:

Regime 1 for $x_{t-1} \leq 0$:

$$x_t = -0.02339012 - 0.18709330x_{t-1} + a_t$$

Regime 2 for $x_{t-1} > 0$:

$$x_t = -0.005189495 - 0.086354193x_{t-1} + 0.209213849x_{t-2} + 0.242253421x_{t-3} + 0.131535088x_{t-4} + a_t$$

e. Compare the linear AR model and the TAR model.

```
arima_model <- arima(unemp_diff, order = c(13, 0, 0))

arima_model$aic
```

```
## [1] 1429.825
```

```
tar_model$AIC
```

```
##    1
## 1419
```

Since the TAR model had a lower AIC than the AR model, it is the better model.

Problem 4.

Suppose that the monthly log returns, in percentages, of a stock follow the following Markov switching model:

$$r_t = 1 + a_t \quad a_t = \sigma_t \varepsilon_t$$
$$\sigma_t^2 = \begin{cases} 0.1a_{t-1}^2 + 0.8\sigma_{t-1}^2 & \text{if} \quad S_t = 1 \\ 5 + 0.1a_{t-1}^2 + 0.5\sigma_{t-1}^2 & \text{if} \quad S_t = 2 \end{cases}$$

where the transition probabilities are:

$$\mathbb{P}(S_t = 2 \mid S_{t-1} = 1) = 0.1\mathbb{P}(S_t = 1 \mid S_{t-1} = 2) = 0.2$$

Suppose that $a_100 = 6, \sigma_{100}^2 = 50$, and $S_{100} = 2$ with probability 1.

a. What is the 1-step-ahead volatility forecast at the forecast origin $t = 100$?

```
# Given values
a_100 <- 6
sigma2_100 <- 50
S_100 <- 2

# Calculating sigma2_101
if (S_100 == 1) {
```

```r
  sigma2_101 <- 0.1 * a_100^2 + 0.8 * sigma2_100
} else if (S_100 == 2) {
  sigma2_101 <- 5 + 0.1 * a_100^2 + 0.5 * sigma2_100
}

# Printing the forecast
print(sigma2_101)
```

```
## [1] 33.6
```

b. If the probability of $S_{100} = 2$ is reduced to 0.9, what is the 1-step-ahead volatility forecast at the forecast origin $t_1 00$?

```r
# Updated probability of S_100 = 2
prob_S100_eq_2 <- 0.9
prob_S100_eq_1 <- 1 - prob_S100_eq_2

# Transition probabilities
P_S2_given_S1 <- 0.1
P_S1_given_S2 <- 0.2

# Forecasting sigma2_101 using weighted average
sigma2_101_S1 <- 0.1 * a_100^2 + 0.8 * sigma2_100  # if S_101 = 1
sigma2_101_S2 <- 5 + 0.1 * a_100^2 + 0.5 * sigma2_100  # if S_101 = 2

# Weighted forecast
sigma2_101_forecast <- (prob_S100_eq_1 * P_S2_given_S1 * sigma2_101_S1) +
                       (prob_S100_eq_2 * P_S1_given_S2 * sigma2_101_S2)

# Printing the forecast
print(sigma2_101_forecast)
```

```
## [1] 6.484
```