ARMA-GARCH

```
setwd("C:/Users/reshed001/Desktop/OMSA_7.26.2020/ISYE 6402 - Time Series/Module 3/HW_New")
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
```

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

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

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(tseries)
library(fGarch)
```

```
## Warning: package 'fGarch' was built under R version 4.0.3
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Warning: package 'timeSeries' was built under R version 4.0.3
```

```
##
## Attaching package: 'timeSeries'
```

```
## The following object is masked from 'package:zoo':
##
##     time<-
```

```
## Loading required package: fBasics
```

```
## Warning: package 'fBasics' was built under R version 4.0.3
```

```
##
## Attaching package: 'fBasics'
```

```
## The following object is masked from 'package:TTR':
##
##     volatility
```

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```
library(TSA)
```

```
## Warning: package 'TSA' was built under R version 4.0.3
```

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

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

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

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

```
library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 4.0.3
```

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

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

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

```
library(ggpubr)
```

```
## Loading required package: ggplot2
```

```
data = read.csv("TSLA.csv")
head(data)
```

```
##        Date Close
## 1  1/1/2012 26.91
## 2  1/8/2012 22.79
## 3 1/15/2012 26.60
## 4 1/22/2012 29.33
## 5 1/29/2012 31.15
## 6  2/5/2012 31.10
```

```
date = as.Date(as.character(data$Date), format="%m/%d/%Y")

data = log(data[,2])

data.ts = ts(data,start=c(2012,1),freq=52)
data.growth = diff(data.ts)
date = date[-1] # remove first date due to diff
```

Question 1: ARIMA(p,d,q) (15 Points)

With the growth rate data (first difference), use the iteration with the AIC metric selecting the minimum AIC (don't select a simpler model with higher AIC), select the best order of ARIMA model (max order 4,1,4).

Plot residual and square residual ACFs and interpret.

Perform and interpret tests for serial correlation and heteroscedasticity on model residuals.

ARIMA Order Selection

```
order_arma = function(ts_data, max_p, max_d, max_q){
  orders = data.frame()
  for (p in 0:max_p){
    for (d in 0:max_d){
      for (q in 0:max_q) {
        possibleError <- tryCatch({
          mod = arima(ts_data, order=c(p,d,q), method="ML")
          current.aic = AIC(mod)
          #current.aic = current.aic-2*(p+q+1)+2*(p+q+1)*n/(n-p-q-2)
          orders<-rbind(orders,c(p,d,q,current.aic))
        },
          error=function(e) e
        )
        if(inherits(possibleError, "error")) next

      }
    }
  }
  names(orders) <- c("p","d","q","AIC")
  return(orders[order(orders$AIC),])
}

max_p = 4
max_q = 4
max_d = 1
ts_data = data.growth

# Observe first best models based on smallest AICs
orders = order_arma(ts_data, max_p, max_d, max_q)
```

```
## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1

## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1

## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
head(orders)
```

```
##    p d q       AIC
## 43 4 0 2 -1003.803
## 25 2 0 4 -1002.094
## 1  0 0 0 -1001.952
## 44 4 0 3 -1001.778
## 35 3 0 4 -1000.316
## 11 1 0 0  -999.979
```

```
# Optimal order
opt_p = orders[1,1] # 4
opt_d = orders[1,2] # 0
opt_q = orders[1,3] # 2

# Fit the optimal model
model_arma = arima(data.growth,order=c(opt_p,opt_d,opt_q), method="ML")

print(paste("Oprtimal p order is: ", opt_p))
```

```
## [1] "Oprtimal p order is:  4"
```

```
print(paste("Oprtimal d order is: ", opt_d))
```

```
## [1] "Oprtimal d order is:  0"
```

```
print(paste("Oprtimal q order is: ", opt_q))
```

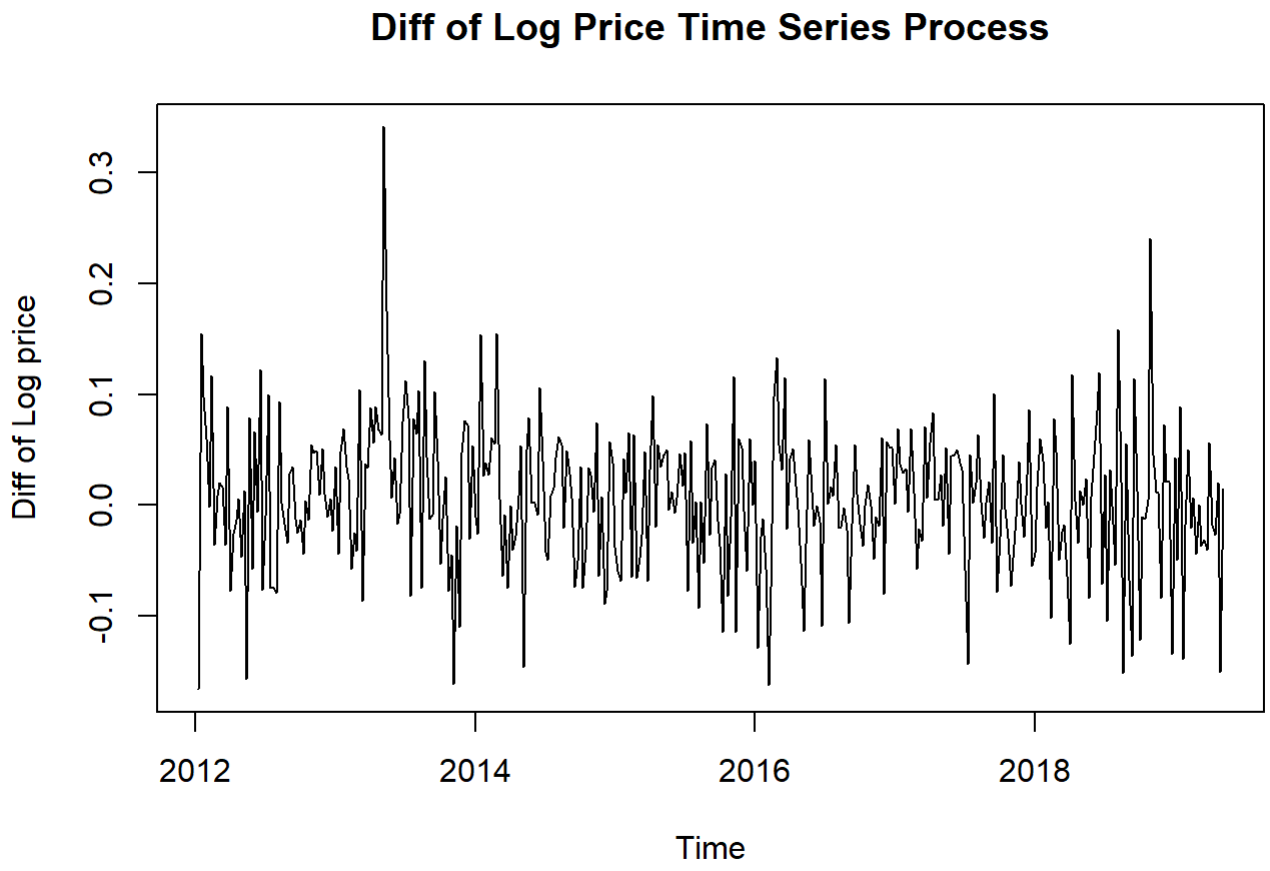
```
## [1] "Oprtimal q order is:  2"
```

Answer

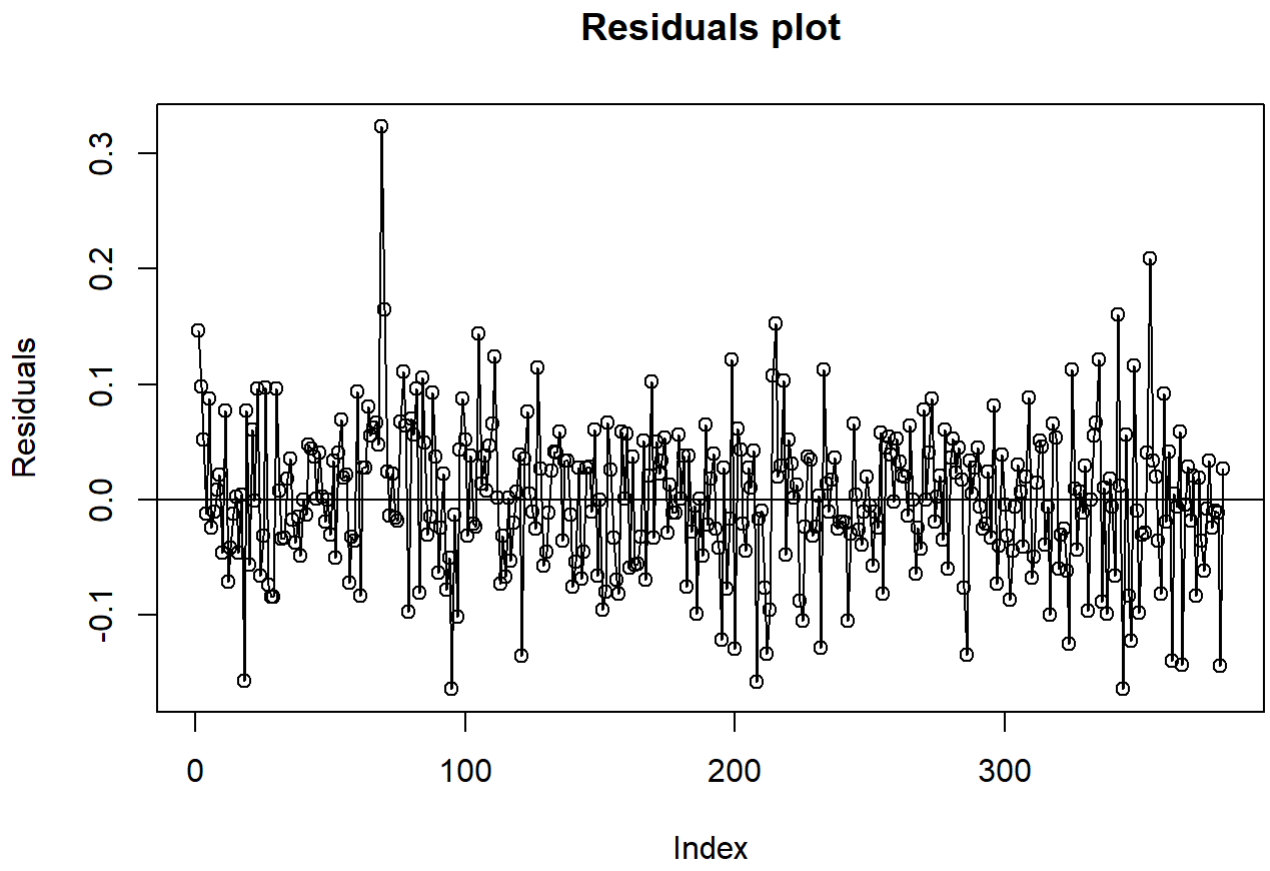
See above the selected ARIMA model is ARIMA(4,0,2) which is the same as ARMA(4,2).

Plot residual and square residual ACFs

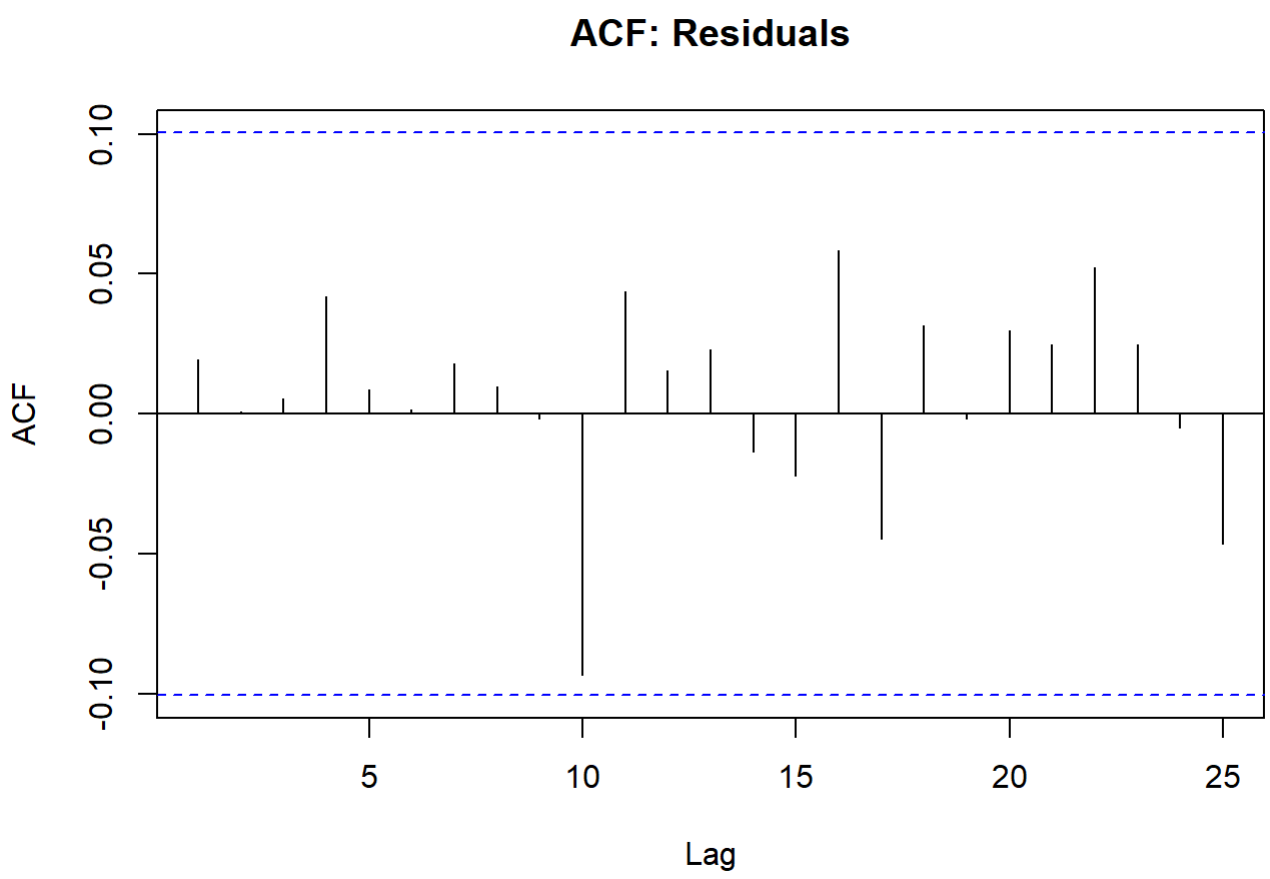
```
# Plot Difference Log price Time Series Process
ts.plot(data.growth, ylab="Diff of Log price", main="Diff of Log Price Time Series Process")
```



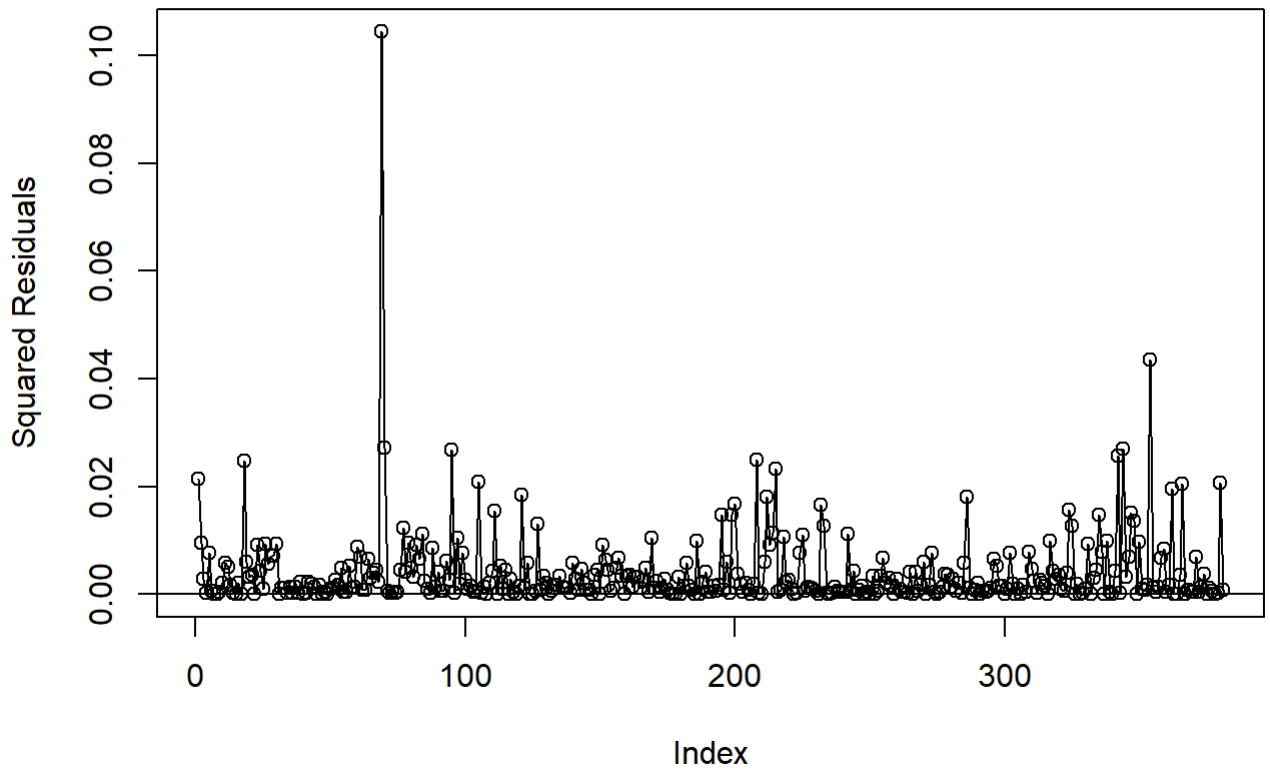

```
res = resid(model_arma)[-1] # remove first data point equals to 0

# Residuals and residuals ACF plots
plot(res, type="o", ylab="Residuals", main="Residuals plot")
abline(h=0)
```



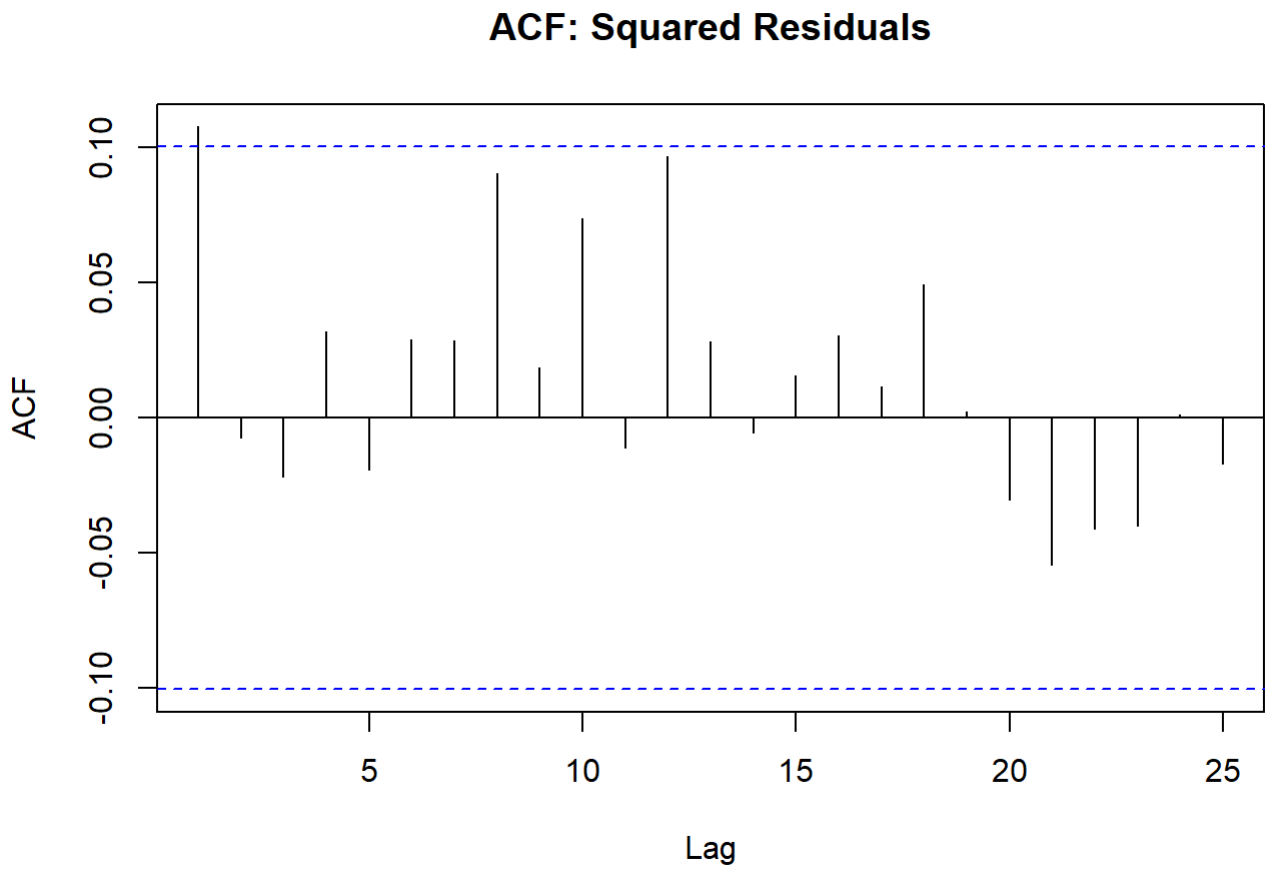

```
acf(res, lag.max=25, main="ACF: Residuals")
```




```
# Squared residuals and squared residuals ACF plots
plot(res^2, type="o", ylab="Squared Residuals", main="Squared Residuals plot")
abline(h=0)
```




```
acf(res^2, lag.max=25, main="ACF: Squared Residuals")
```



Answer

Looking at the “original” time series which is the differenced log price (plot named “Diff of Log Time Series Process”) we can see a constant mean but it seems like the CONDITIONAL variance is not constant which implies heteroscedasticity. We can see large volatility around year 2013 and 2019, and some increase in volatility around year 2016. Since we have time-varied volatility, we would need to analyze the ARMA model’s residuals and transformed residuals (i.e. squared residuals) for serial correlation. Looking at the Residuals ACF plot we see the residuals do not indicate auto-correlation since the values (up to lag 25) are within the significance bands. However, if we look at the Squared Residuals ACF plot, we can see an indication of serial correlation as the auto-correlation value at lag 1 is outside the significance bands, and hence an idication of heteroscedasticity. It is good practice to evaluate for uncorrelation using hypothesis testing in addition to visual analysis. So let us perform the Box-Ljung test below.

Tests for serial correlation and heteroscedasticity

```
# test for serial correlation in residuals
Box.test(res, lag=opt_p+opt_q+1, type='Ljung',fitdf=opt_p+opt_q)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 0.987, df = 1, p-value = 0.3205
```

```
# test for serial correlation in squared residuals
Box.test(res^2, lag=opt_p+opt_q+1, type='Ljung',fitdf=opt_p+opt_q)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 5.8395, df = 1, p-value = 0.01567
```

Answer

The Box-Ljung's Null hypothesis is that the residuals are uncorrelated. With a p-value 0.3205, we cannot reject the null hypothesis (at the α = 0.05 level) and hence, we conclude that the residuals are uncorrelated. However, with a p-value of 0.01567, we reject the null hypothesis and conclude the squared residuals are correlated. ***Hence, based on the Box-Ljung test, the residuals are uncorrelated but not independent since the squared residuals are correlated***, and we conclude we have heteroscedasticity, similar to our conclusion above.

Question 2: ARMA(p,q)-GARCH(m,n) (20 Points)

With the growth rate data (first difference) and using the multi-step refinement method (see piazza post @280) with minimum BIC starting with ARMA order calculated in Q1, find the best ARMA(P,Q)-GARCH(M,N) order pair (max GARCH order 3,3). Fit this model.

Fully write out the equation. You can simply state the level of differencing Y represents in the data for simplicity sake.

Perform goodness of fit tests on this model and interpret. (Hint: summary of a garchFit model performs them all, you just need to interpret).

```
# Initial GARCH Order
# ARIMA-GARCH: Select GARCH order
# Initial ARMA(4,2)
# Max GARCH order (3,3)

test_modelAGG <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(4,2),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, data.growth, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(m,n,current.bic)
    names(df) <- c("m","n","BIC")
    print(paste(m,n,current.bic,sep=" "))
    return(df)
}

ordersAGG = data.frame(Inf,Inf,Inf)
names(ordersAGG) <- c("m","n","BIC")

for (m in 0:3){
    for (n in 0:3){
        possibleError <- tryCatch(
            ordersAGG<-rbind(ordersAGG,test_modelAGG(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
```

```
## [1] "0 0 -2.56390336438806"
## [1] "0 1 -2.54828047735607"
## [1] "0 2 -2.52640253908174"
## [1] "0 3 -2.51717388475447"
## [1] "1 0 -2.44543643661524"
## [1] "1 1 -2.54946776356549"
## [1] "1 2 -2.58530066319158"
## [1] "1 3 -2.58614144586915"
## [1] "2 0 -2.53658372933802"
## [1] "2 1 -2.53390384643259"
## [1] "2 2 -2.51887509720818"
## [1] "2 3 -2.56877973208704"
## [1] "3 0 -2.50784345663187"
## [1] "3 1 -2.57499071854611"
## [1] "3 2 -2.55572211886978"
## [1] "3 3 -2.55683765724558"
```

```
ordersAGG <- ordersAGG[order(-ordersAGG$BIC),]
tail(ordersAGG)
```

```
##    m n       BIC
## 17 3 3 -2.556838
## 2  0 0 -2.563903
## 13 2 3 -2.568780
## 15 3 1 -2.574991
## 8  1 2 -2.585301
## 9  1 3 -2.586141
```

GARCH(1,3)

```
# ARMA update
# ARIMA-GARCH: Select ARIMA order
# Max ARMA order (4,4)

test_modelAGA <- function(p,q){
    spec = ugarchspec(variance.model=list(garchOrder=c(1,3)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, data.growth, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(p,q,current.bic)
    names(df) <- c("p","q","BIC")
    print(paste(p,q,current.bic,sep=" "))
    return(df)
}

ordersAGA = data.frame(Inf,Inf,Inf)
names(ordersAGA) <- c("p","q","BIC")
for (p in 0:4){
    for (q in 0:4){
        possibleError <- tryCatch(
            ordersAGA<-rbind(ordersAGA,test_modelAGA(p,q)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
```

```
## [1] "0 0 -2.58397060138191"
## [1] "0 1 -2.5690970307804"
## [1] "0 2 -2.55453618381674"
## [1] "0 3 -2.53914769890282"
## [1] "0 4 -2.53058023137355"
## [1] "1 0 -2.56915169555075"
## [1] "1 1 -2.55726242932836"
## [1] "1 2 -2.54198813263015"
## [1] "1 3 -2.5268975978481"
## [1] "1 4 -2.52426342653738"
## [1] "2 0 -2.55410280972165"
## [1] "2 1 -2.53887935632634"
## [1] "2 2 -2.54925085559696"
## [1] "2 3 -2.5687482770954"
## [1] "2 4 -2.51839919123002"
## [1] "3 0 -2.53975699297666"
## [1] "3 1 -2.52534822284405"
## [1] "3 2 -2.5116127099138"
## [1] "3 3 -2.50947148147104"
## [1] "3 4 -2.50398310989308"
## [1] "4 0 -2.53118923211884"
## [1] "4 1 -2.52039448099236"
## [1] "4 2 -2.58614144586915"
## [1] "4 3 -2.57789023732717"
## [1] "4 4 -2.54510585514929"
```

```
ordersAGA <- ordersAGA[order(-ordersAGA$BIC),]
tail(ordersAGA)
```

```
##    p q       BIC
## 15 2 3 -2.568748
## 3  0 1 -2.569097
## 7  1 0 -2.569152
## 25 4 3 -2.577890
## 2  0 0 -2.583971
## 24 4 2 -2.586141
```

ARMA(4,2)

```
#GARCH update
# Max GARCH order (3,3)

test_modelAGG <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                      mean.model=list(armaOrder=c(4,2),
                                      include.mean=T), distribution.model="std")
    fit = ugarchfit(spec, data.growth, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(m,n,current.bic)
    names(df) <- c("m","n","BIC")
    print(paste(m,n,current.bic,sep=" "))
    return(df)
}

ordersAGG = data.frame(Inf,Inf,Inf)
names(ordersAGG) <- c("m","n","BIC")

for (m in 0:3){
    for (n in 0:3){
        possibleError <- tryCatch(
            ordersAGG<-rbind(ordersAGG,test_modelAGG(m,n)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
```

```
## [1] "0 0 -2.56390336438806"
## [1] "0 1 -2.54828047735607"
## [1] "0 2 -2.52640253908174"
## [1] "0 3 -2.51717388475447"
## [1] "1 0 -2.44543643661524"
## [1] "1 1 -2.54946776356549"
## [1] "1 2 -2.58530066319158"
## [1] "1 3 -2.58614144586915"
## [1] "2 0 -2.53658372933802"
## [1] "2 1 -2.53390384643259"
## [1] "2 2 -2.51887509720818"
## [1] "2 3 -2.56877973208704"
## [1] "3 0 -2.50784345663187"
## [1] "3 1 -2.57499071854611"
## [1] "3 2 -2.55572211886978"
## [1] "3 3 -2.55683765724558"
```

```
ordersAGG <- ordersAGG[order(-ordersAGG$BIC),]
tail(ordersAGG)
```

```
##    m n       BIC
## 17 3 3 -2.556838
## 2  0 0 -2.563903
## 13 2 3 -2.568780
## 15 3 1 -2.574991
## 8  1 2 -2.585301
## 9  1 3 -2.586141
```

GARCH(1,3)

ARMA(4,2)-GARCH(1,3)

```
arch_garch = garchFit(~ arma(4,2)+garch(1,3), data = data.growth, trace = FALSE)
```

```
## Warning in sqrt(diag(fit$cvar)): NaNs produced
```

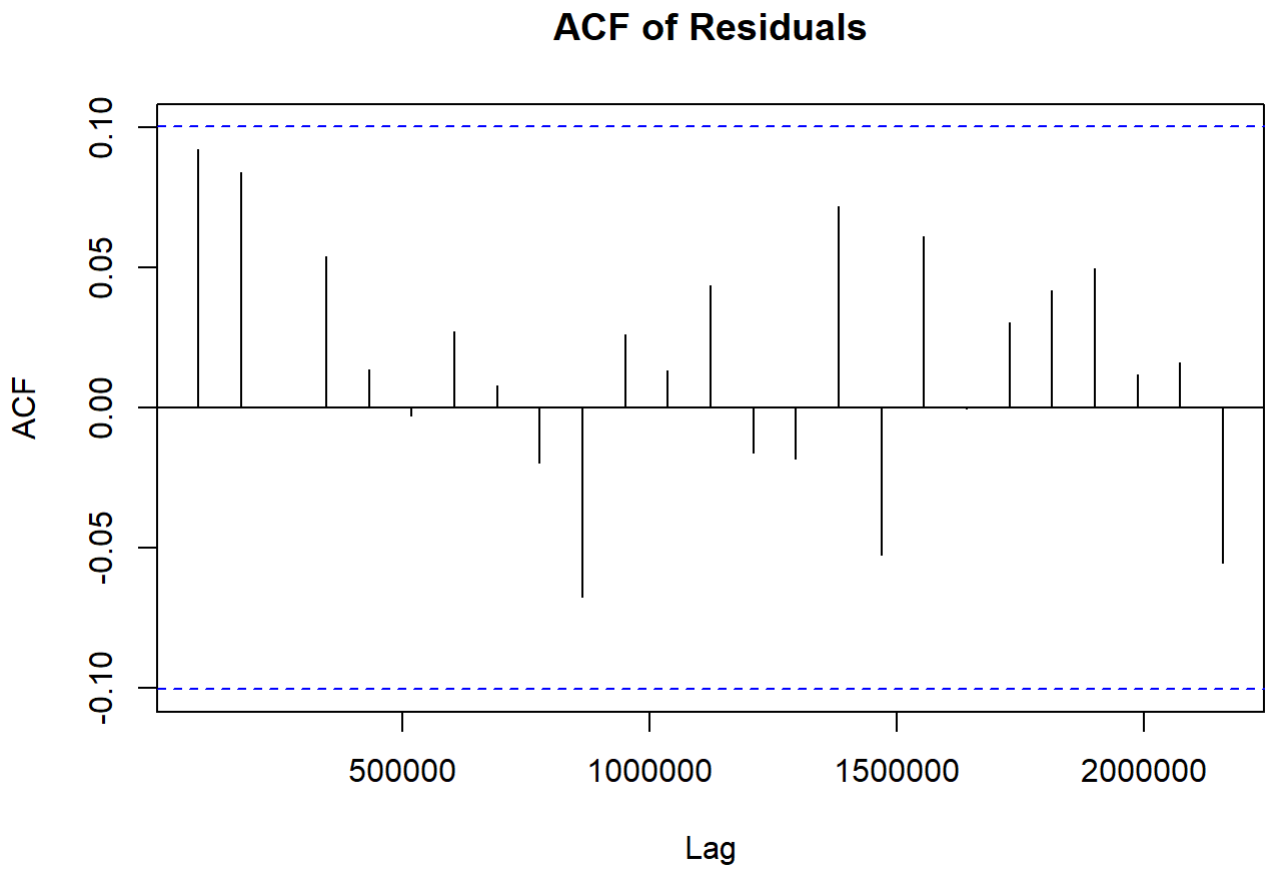
```
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```

```
summary(arch_garch)
```

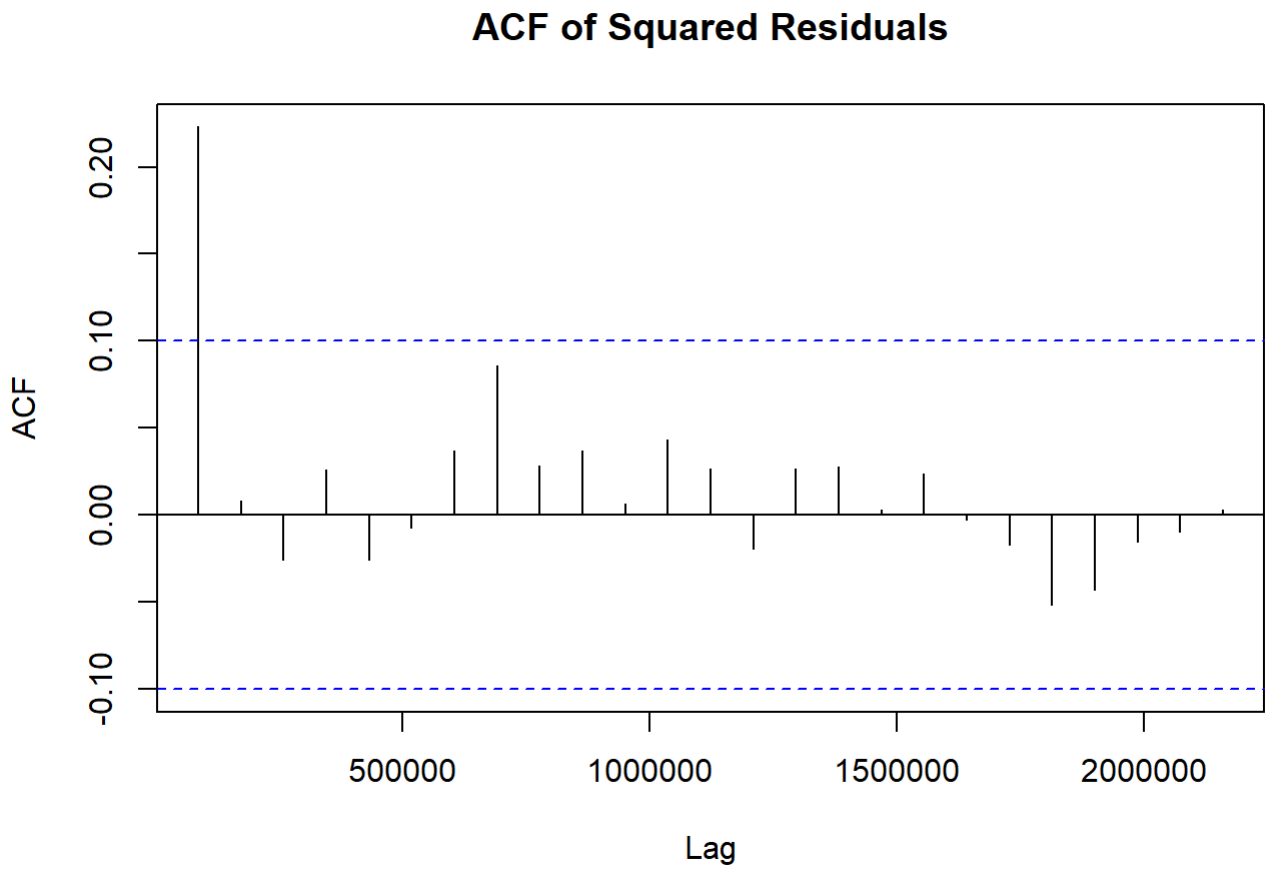
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(4, 2) + garch(1, 3), data = data.growth,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(4, 2) + garch(1, 3)
## <environment: 0x00000000232aa6f0>
##  [data = data.growth]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##          mu          ar1          ar2          ar3          ar4          ma1
##  0.00997014  -0.98370719  -0.48917475   0.08526284   0.12868714   0.99589813
##         ma2        omega       alpha1        beta1        beta2        beta3
##  0.55714540   0.00060916   0.09395319   0.00000001   0.15904131   0.59571055
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      9.970e-03   7.237e-03    1.378 0.168286
## ar1    -9.837e-01   1.289e-01   -7.630 2.35e-14 ***
## ar2    -4.892e-01   1.366e-01   -3.581 0.000343 ***
## ar3     8.526e-02   8.022e-02    1.063 0.287838
## ar4     1.287e-01   5.229e-02    2.461 0.013858 *
## ma1     9.959e-01   1.312e-01    7.591 3.18e-14 ***
## ma2     5.571e-01   1.581e-01    3.525 0.000423 ***
## omega   6.092e-04   3.500e-04    1.741 0.081770 .
## alpha1  9.395e-02   5.035e-02    1.866 0.062051 .
## beta1   1.000e-08          NA       NA       NA
## beta2   1.590e-01          NA       NA       NA
## beta3   5.957e-01          NA       NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  519.0287    normalized:  1.358714
##
## Description:
##  Thu Mar 25 16:04:09 2021 by user: reshed001
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  80.57813  0
##  Shapiro-Wilk Test  R    W      0.9788229 2.173276e-05
```

```
##  Ljung-Box Test     R    Q(10)  7.269197  0.6998101
##  Ljung-Box Test     R    Q(15)  8.710996  0.8921373
##  Ljung-Box Test     R    Q(20)  10.26251  0.9631582
##  Ljung-Box Test     R^2  Q(10)  3.830814  0.9546566
##  Ljung-Box Test     R^2  Q(15)  5.553774  0.9863089
##  Ljung-Box Test     R^2  Q(20)  6.608866  0.9977845
##  LM Arch Test       R    TR^2   5.217548  0.950315
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -2.654601 -2.530661 -2.656495 -2.605431
```

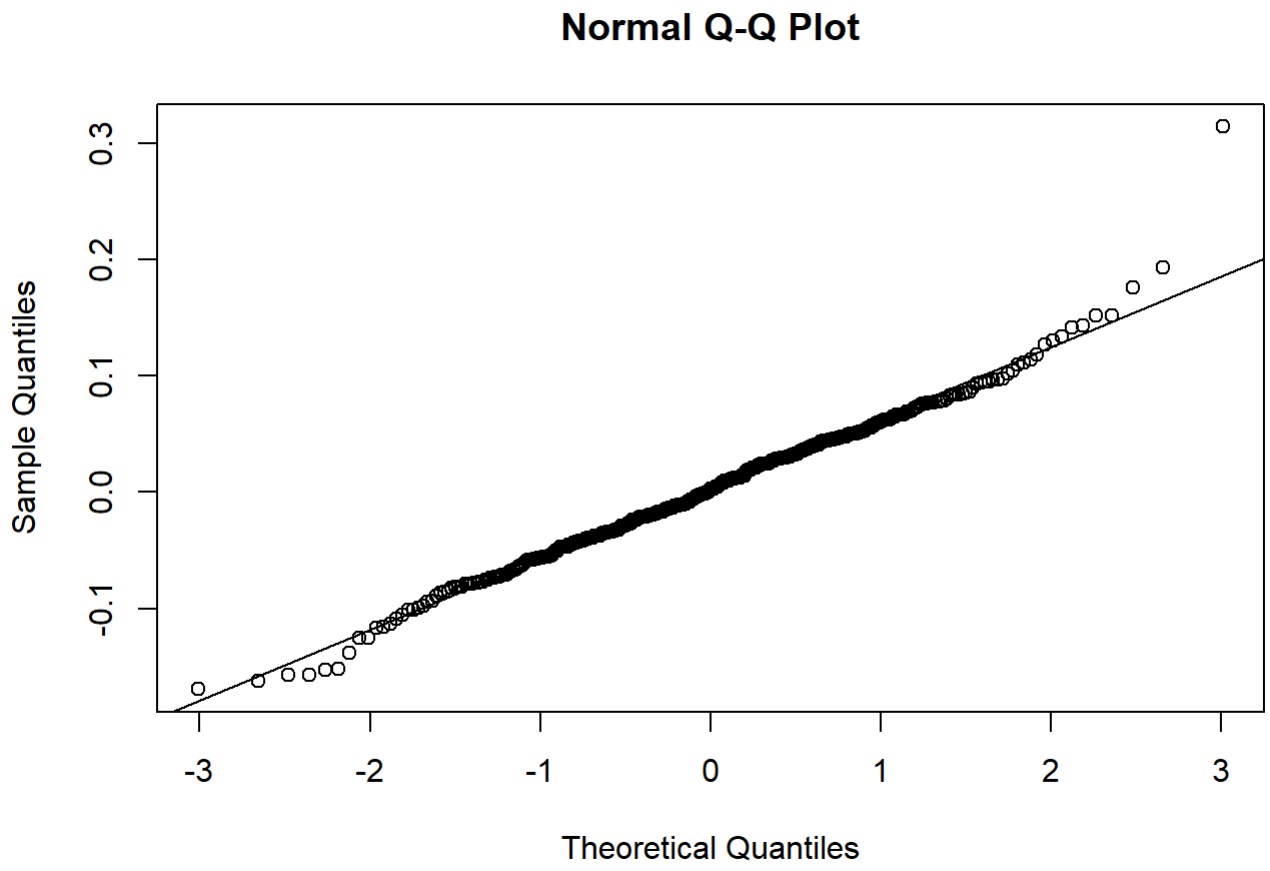
```
## Residual Analysis

spec = ugarchspec(variance.model=list(garchOrder=c(1,3)),
            mean.model=list(armaOrder=c(4,2),
            include.mean=T), distribution.model="std")
arch_garch_model = ugarchfit(spec, data.growth, solver = 'hybrid')

res_final = residuals(arch_garch_model)

acf(res_final, main="ACF of Residuals")
```



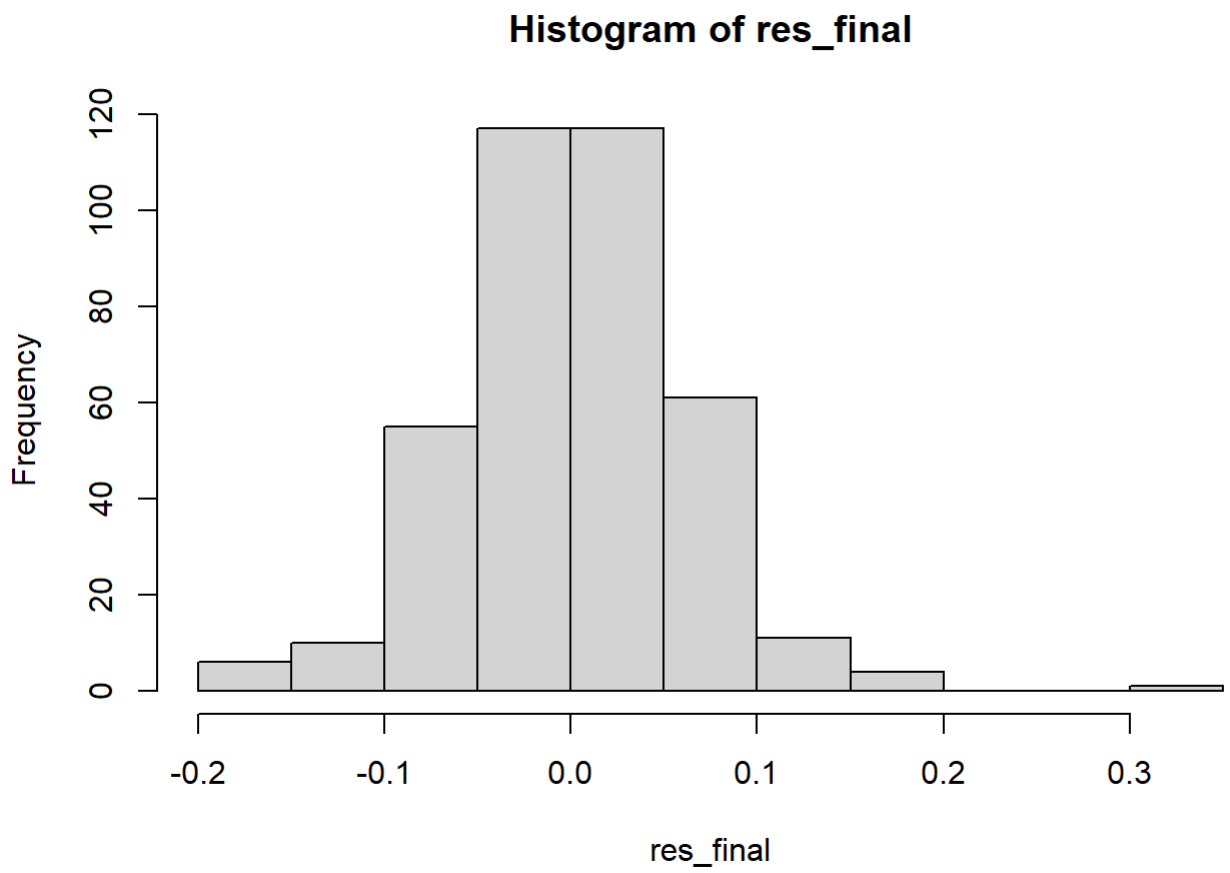

```
acf(res_final^2, main="ACF of Squared Residuals")
```



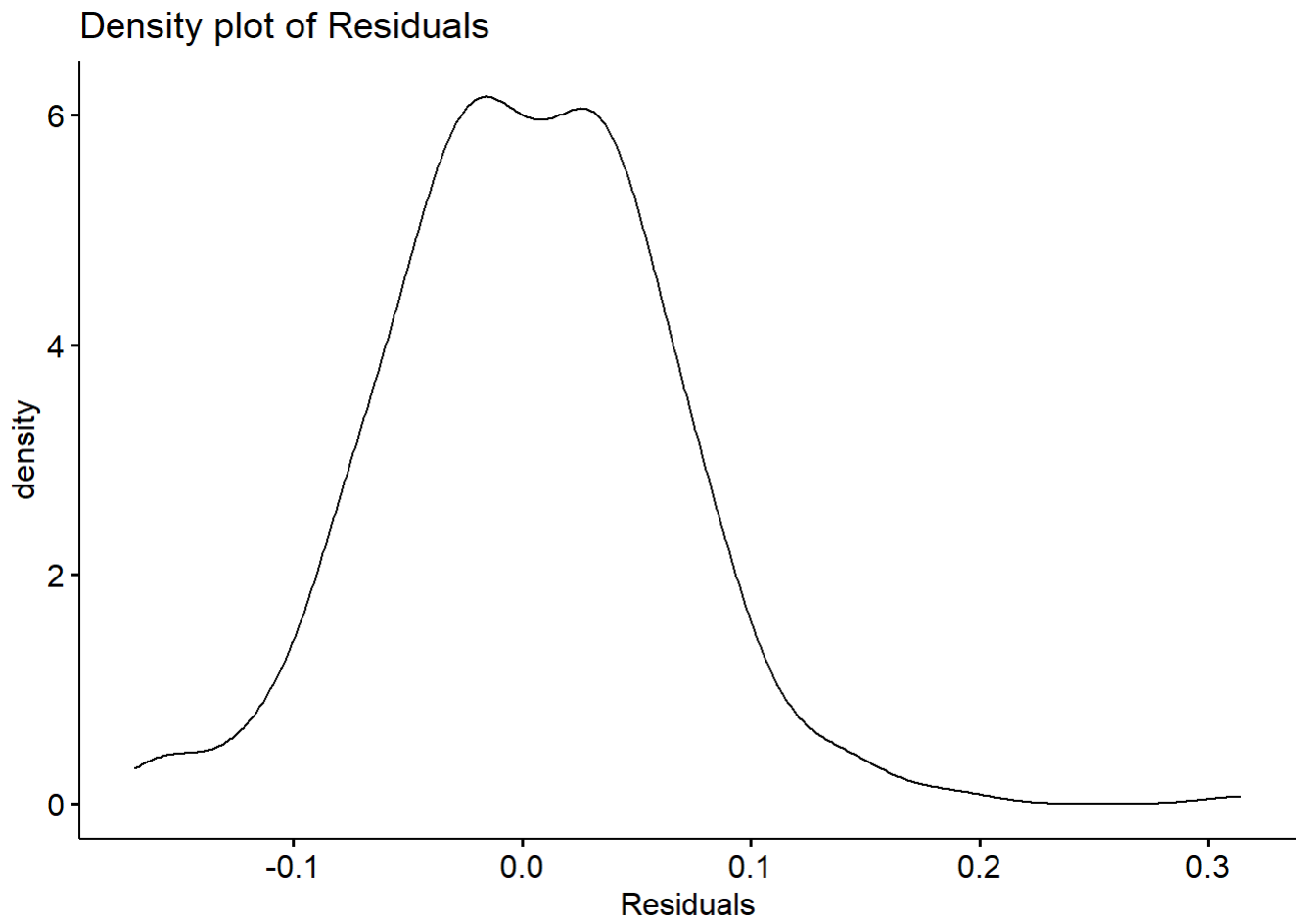

```
qqnorm(res_final)
qqline(res_final)
```




```
hist(res_final)
```




```
ggdensity(res_final,
          main = "Density plot of Residuals",
          xlab = "Residuals")
```



Answer

Best ARMA(p,q)-GARCH(m,n) is ARMA(4,2)-GARCH(1,3).

Given Y_t = Δlog(price), the model's equations are:

ARMA portion:

Y_t = 0.00997 - 0.9837Y_{t-1} - 0.4892Y_{t-2} + 0.0852Y_{t-3} + 0.1287Y_{t-4} + 0.9959Z_{t-1} + 0.5571Z_{t-2} + Z_t

GARCH portion:

(Z_t | F_{t-1}) $\sim$ N(0, σ_t^2)

σ_t^2 = 0.000609 + 0.09395Z_{t-1}^2 + 0.00000001σ_{t-1}^2 + 0.1590σ_{t-2}^2 + 0.5957σ_{t-3}^2

Goodness of Fit/ Residuals Analysis

Looking at the ACF plot of the residuals, all the values are within the confidence bands. The ACF plot of the squared residuals indicates a white noise since all of the values are within the confidence band other than lag 0. Hence, we can conclude that there is no auto-correlation in the residuals and squared residuals processes. Similarly, the various Ljung-Box tests results in the model's output indicate the residuals and squared residuals do not show serial correlation. All of the Ljung-Box tests' show large p-value, meaning we ***cannot*** reject the null hypothesis the residuals/ squared residuals are uncorrelated (hence they are uncorrelated).

The Q-Q Normal and Histogram plots of the residuals show tailed distribution, but are close to be normal distributed. However, looking at both the Jarque-Bera and Shapiro-Wilk tests result in the model output, with a small p-value of ~ 0, we conclude the residuals are not normal distributed. Note that the Jarque-Bera and Shapiro-

Wilk tests' null hypothesis is that the distribution is normal and the ~ 0 p-value indicates we can reject the null which means non-normal distribution. The Density plot of residuals also indicate non-normality and heavy tailed. However, the LM Arch test's null is that "the squared residuals are a sequence of white noise, namely, the residuals are homoscedastic." With a large p-value of 0.950315, we cannot reject the null and hence we conclude the model is a good fit (note that these are the residuals of the ARMA-GARCH model and if these residuals are homoscedastic, it means we captured the entire GARCH effect in our model).

Question 3: Forecasting (15 Points)

Using the rolling forecasting method (for each pred, fit A-G model to all points prior), forecast last 40 points.

Calculate MAPE, and Precision

Overlay the draw an overlay of the predicted points on the original series in that range.

```
# Split data to train and test sets
n_predict = 40
n=length(data.growth)
nfit = n - n_predict

train = data.growth[1:nfit]
test = tail(data.growth, n_predict)
```

Forecast

```
spec = ugarchspec(variance.model=list(garchOrder=c(1,3)),
                  mean.model=list(armaOrder=c(4,2),
                  include.mean=T), distribution.model="std")

nfore = length(test)
fore.series = NULL
fore.sigma = NULL

for(f in 1: nfore){
    ## Fit models
    data = train
    if(f>=2)
      data = c(train,test[1:(f-1)])
    final.model = ugarchfit(spec, data, solver = 'hybrid')

  ## Forecast
    fore = ugarchforecast(final.model, n.ahead=1)
    fore.series = c(fore.series, fore@forecast$seriesFor)
    fore.sigma = c(fore.sigma, fore@forecast$sigmaFor)
}
```

Accuracy Measures

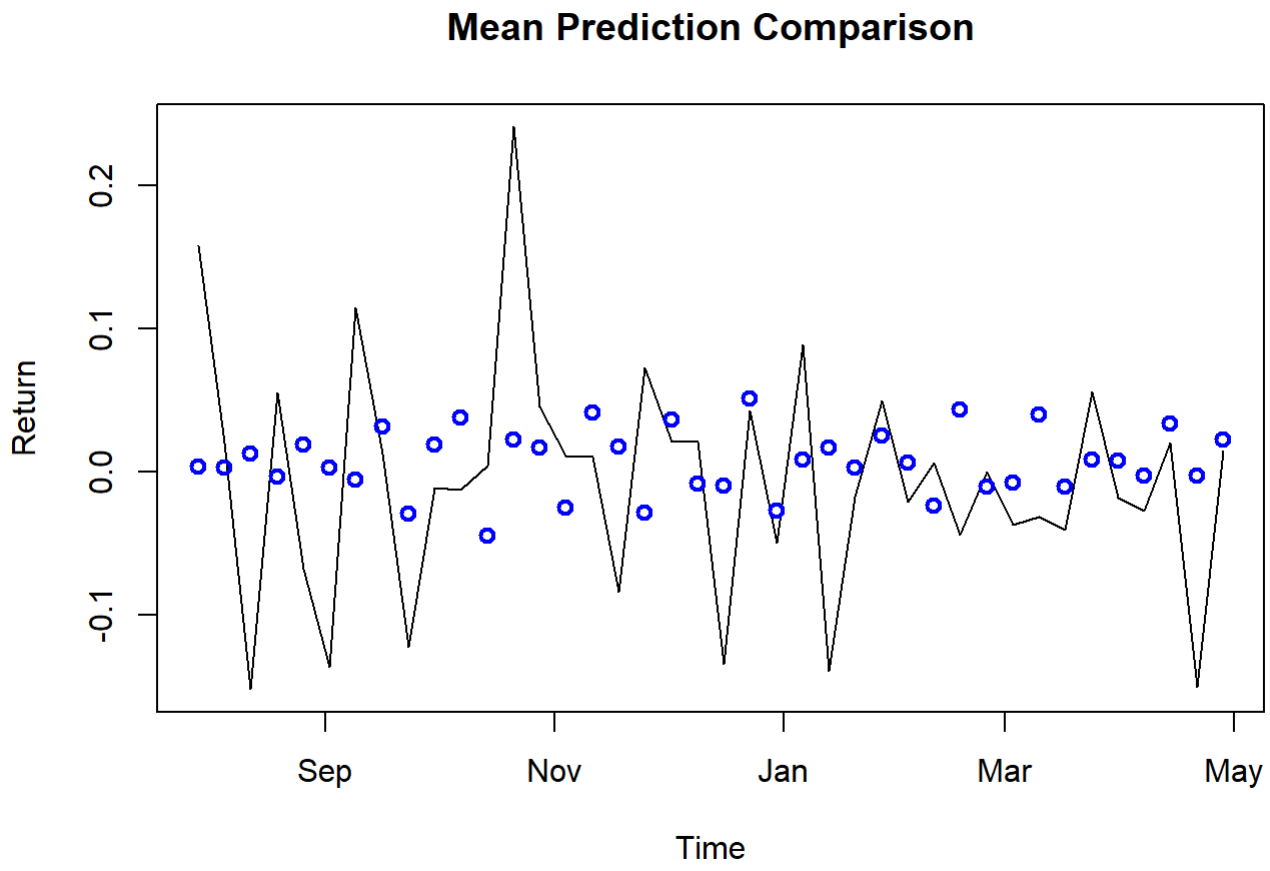
```
#Compute Accuracy Measures

#Mean Squared Prediction Error (MSPE)
#mean((fore.series - test)^2)

#Mean Absolute Prediction Error (MAE)
#mean(abs(fore.series - test))

#Mean Absolute Percentage Error (MAPE)
MAPE = mean(abs(fore.series - test)/abs(test))

#Precision Measure (PM)
PM = sum((fore.series - test)^2)/sum((test-mean(test))^2)

print(paste("Mean Absolute Percentage Error (MAPE) is: ", MAPE))
```

```
## [1] "Mean Absolute Percentage Error (MAPE) is:  2.53034611800942"
```

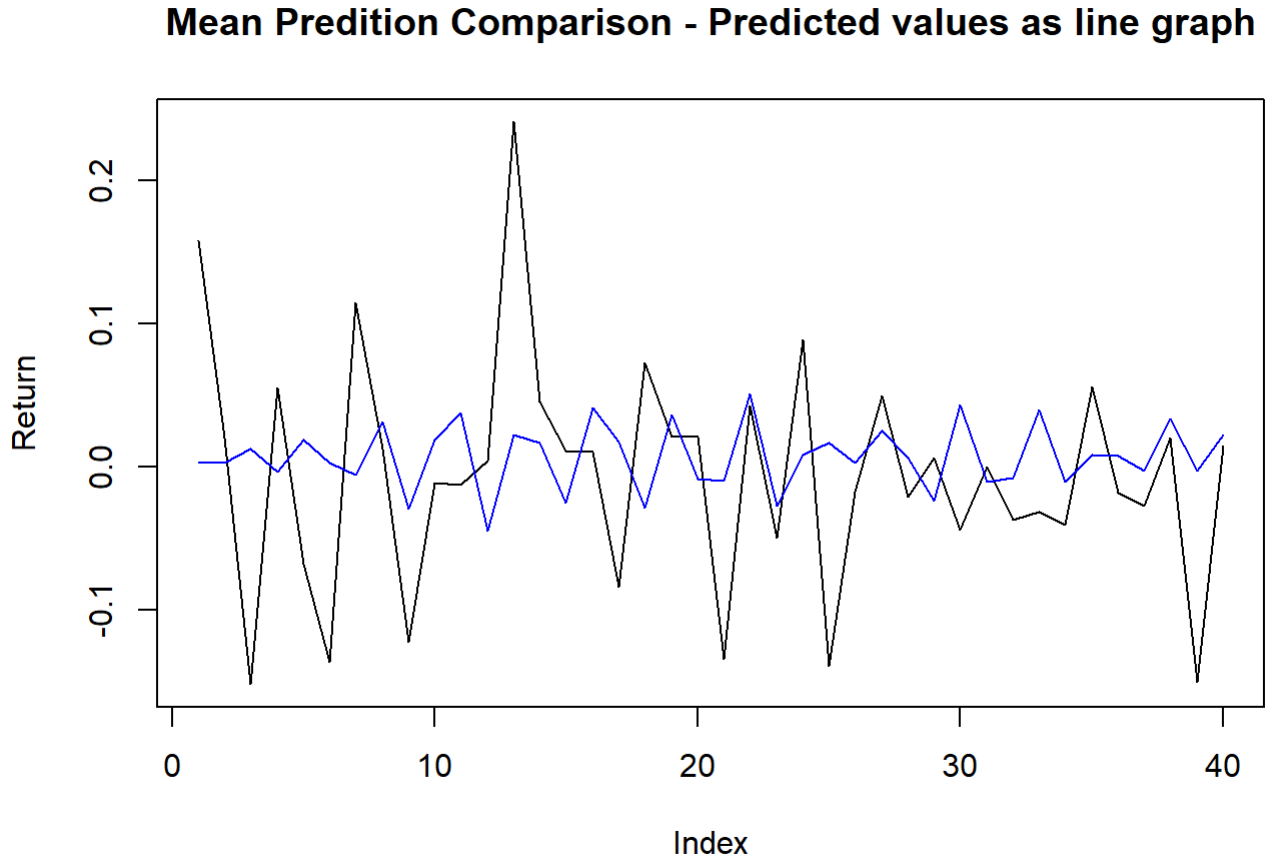
```
print(paste("Precision Measure (PM) is: ", PM))
```

```
## [1] "Precision Measure (PM) is:  1.04400351596905"
```

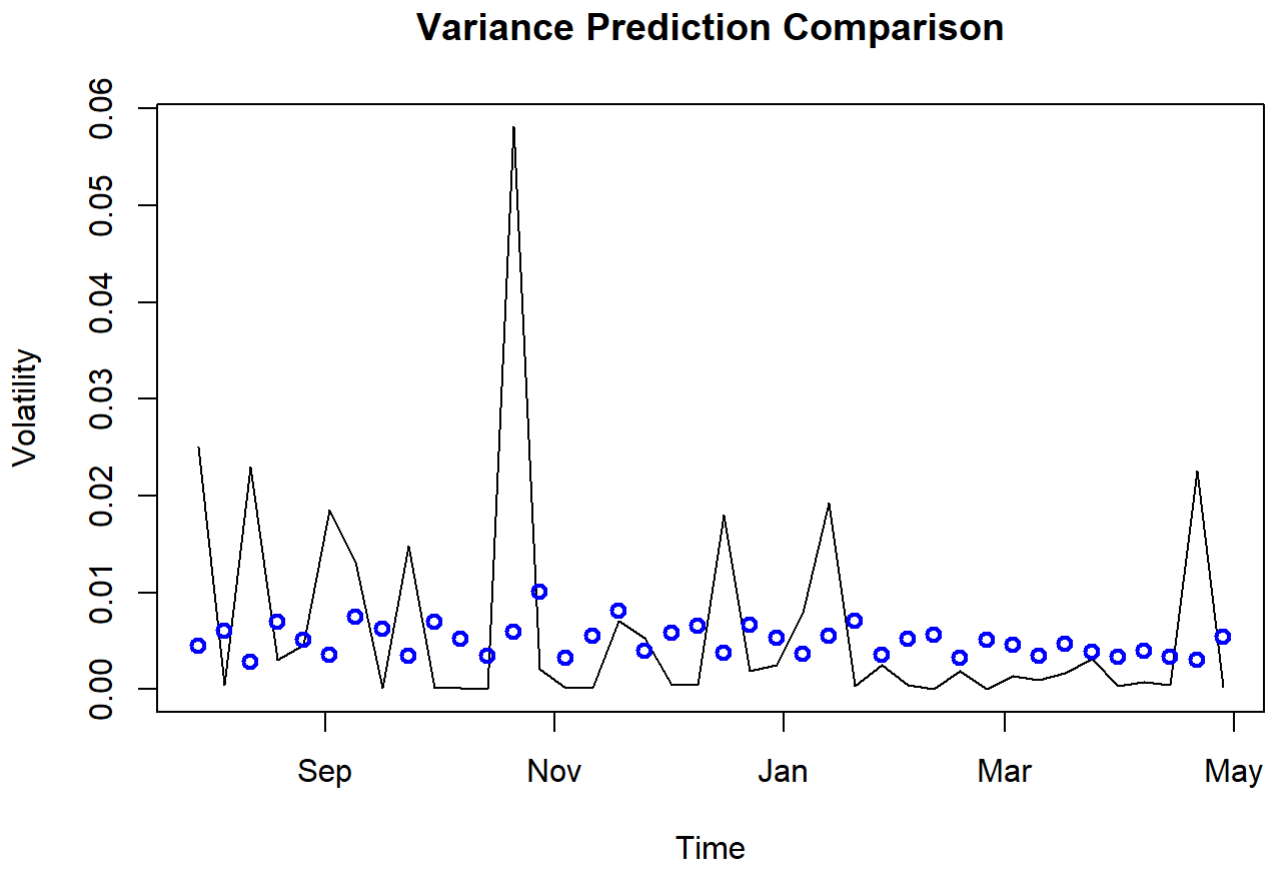
```
# Plot Conditional Mean
ymin = min(c(as.vector(test),fore.series), na.rm = T)
ymax = max(c(as.vector(test),fore.series), na.rm = T)

plot(tail(date, n_predict), test, type="l", ylim=c(ymin,ymax), xlab="Time", ylab="Return", main=
"Mean Prediction Comparison")
points(tail(date, n_predict), fore.series,lwd= 2, col="blue")
```




```
# Another quick way to plot conditional mean
ts.plot(test, xlab="Index", ylab="Return", main="Mean Predition Comparison - Predicted values as
line graph")
lines(fore.series, col="blue")
```




```
################################################################################
# Plot Conditional Variance
ymin = min(c(as.vector(test^2),fore.sigma^2), na.rm = T)
ymax = max(c(as.vector(test^2),fore.sigma^2), na.rm = T)

plot(tail(date, n_predict), test^2, type="l", ylim=c(ymin,ymax), xlab="Time", ylab="Volatility",
main="Variance Prediction Comparison")
points(tail(date, n_predict), fore.sigma^2,lwd= 2, col="blue")
```



Answer

MAPE, calculated as the mean of the absolute values of the differences between the foretasted and actual values, scaled by the actual responses is 2.53. The PM is 1.044, indicates that the proportion between the variability of the prediction versus the variability of the testing data is 1.044. Thus, the variability prediction is almost similar to that in the data. Looking at the Mean Prediction Comparison plot, the predicted mean are around 0 and the model captures the variation in the observed data to some extent but not perfectly. This is the case when the volatility of the time series process predominates over the conditional mean or accounts for the variability to some extent not captured by the mean prediction. Looking at the Conditional Variance Comparison plot, we can see the model did not capture the volatility very well. The model under-predicted the volatility for some points (e.g. September) and over-predicted the volatility for other points (e.g. from February to April).

Thank You!