npark62 4/10/20

ISYE6402 HW4 Peer Response

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).

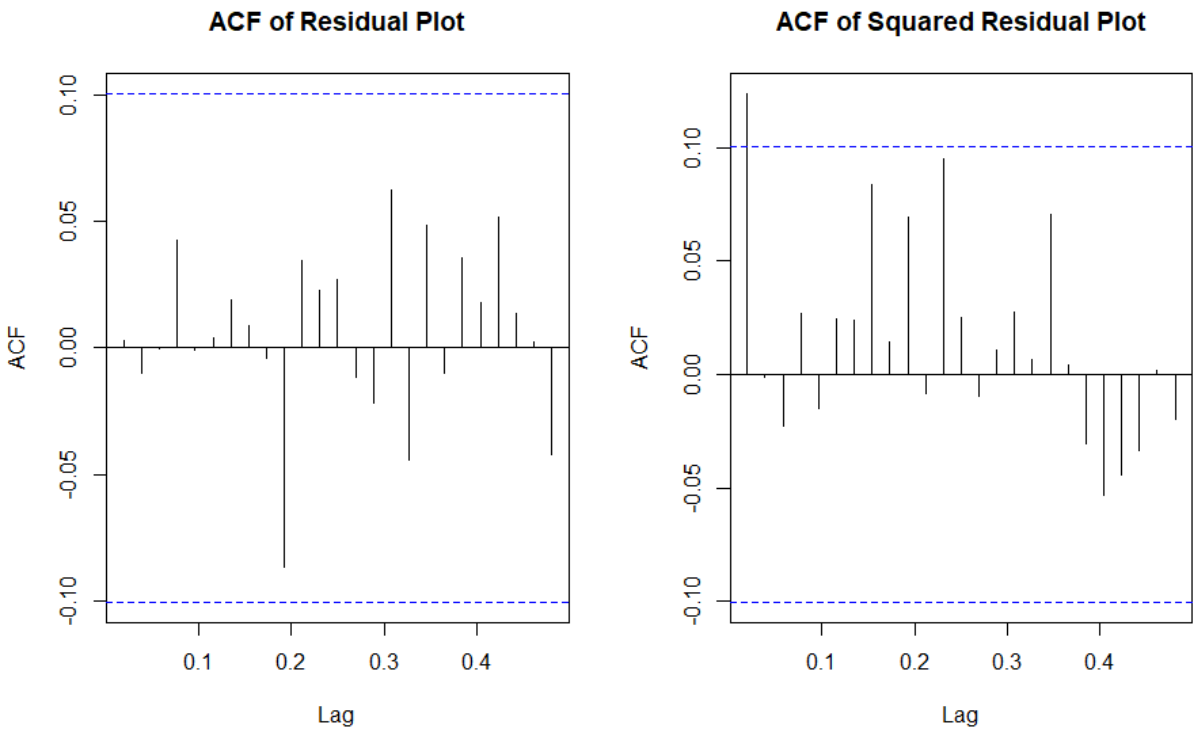
Plot residual and square residual ACFs and interpret.

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

```
# p d q       AIC
# 1 0 0  -999.979
# 3 0 4 -1000.316
# 4 0 3 -1001.778
# 0 0 0 -1001.952
# 2 0 4 -1002.094
# 4 0 2 -1003.803

# Applying no threshold, we choose ARIMA(4,0,2) due to minimum AIC

# Coefficients:
#          ar1      ar2     ar3     ar4     ma1     ma2  intercept
#         -1.2930  -0.8478  0.1053  0.1200  1.3200  0.9343     0.0058
# s.e.   0.0662   0.1212  0.0848  0.0573  0.0446  0.0928     0.0036
# sigma^2 estimated as 0.00405:  log likelihood = 509.9,  aic = -1005.8
```



Both residual and squared residual ACFs somewhat resemble white noise.
Squared residual ACF does not resemble white noise perfectly on higher lag.max which might imply uncorrelation + dependence.

Box.test(resids, lag = 6, type = 'Ljung', fitdf = 5)
# lag number = p order + q order = 6
# Box-Ljung test
#
data: resids
X-squared = 0.74572, df = 1, p-value = 0.3878

p-value is high so we accept the null of uncorrelated residual.

Box.test(resids^2, lag = 6, type = 'Ljung', fitdf = 5)
# Box-Ljung test
#
data: resids^2
X-squared = 6.6919, df = 1, p-value = 0.009685

p-value is lower than an alpha of 0.01 so we reject the null of uncorrelated squared residuals.

Because squared residuals are correlated, the variance is non-constant. (i.e. heteroskedastic).

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 @392) with minimum BIC starting with ARMA order calculated in Q1, find the best ARMA(P,Q)-GARCH(M,N) order pair. 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).

```
# m n       BIC
# 2 1 -2.533904
# 2 0 -2.536584
# 0 1 -2.548280
# 1 1 -2.549468
# 0 0 -2.563903
# 1 2 -2.585301

# With ARMA(4,2), minimum BIC occurs when garch(m = 1, n = 2)

# First model produced, then, is ARMA(4,2)-GARCH(1,2)

# p q       BIC
# 0 1 -2.579060
# 1 0 -2.579097
# 4 3 -2.581914
# 4 2 -2.585301
# 0 0 -2.594143
# 4 4 -2.631685

# Lowest BIC, when GARCH(1,2), is when ARMA(p=4,q=4)
# Second model produced, then, is ARMA(4,4)-GARCH(1,2)

# m n       BIC
# 0 1 -2.518347
# 2 2 -2.531000
# 0 2 -2.545794
# 1 0 -2.559581
# 1 2 -2.631685
# 1 1 -2.650102
```

```
# Lowest BIC, when ARMA(4,4), is when GARCH(1,1)
# Third produced model, then is, ARMA(4,4)-GARCH(1,1)

# final.model.1 = ARMA(4,2) - GARCH(1,2)
# final.model.2 = ARMA(4,4) - GARCH(1,2)
# final.model.3 = ARMA(4,4) - GARCH(1,1)

infocriteria(final.model.1) # BIC: -2.585301
infocriteria(final.model.2) # BIC: -2.631685
infocriteria(final.model.3) # BIC: -2.650102 ***lowest BIC

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

# ARMA(4,4) - GARCH(1,1)

# Optimal Parameters
# ------------------------------------
#   Estimate  Std. Error     t value Pr(>|t|)
# mu      0.002996    0.000031    96.12606 0.000000
# ar1    -1.580124    0.000226 -6987.71919 0.000000
# ar2    -1.565839    0.000215 -7273.18370 0.000000
# ar3    -1.561086    0.000210 -7418.86845 0.000000
# ar4    -0.961903    0.000166 -5781.61309 0.000000
# ma1     1.583024    0.000308  5140.08970 0.000000
# ma2     1.553664    0.000303  5127.27566 0.000000
# ma3     1.611279    0.000364  4420.77485 0.000000
# ma4     1.070483    0.000113  9504.65001 0.000000
# omega   0.000289    0.000125     2.31412 0.020661
# alpha1  0.055345    0.060811     0.91012 0.362757
# beta1   0.859849    0.107686     7.98474 0.000000
# shape   9.634077    3.624748     2.65786 0.007864

# Y_t = 0.002996  - 1.580124*Y_(t-1) -1.565839*Y_(t-2) -1.561086*Y_(t-3) - 0.961903*Y_(t-
4) + Z_t + 1.583024*Z_(t-1)+1.553664*Z_(t-2) +1.611279*Z_(t-3) + 1.070483*Z_(t-4)
# Sigma^2_t = 0.000289 + 0.055345*Z^2_(t-1) + 0.859849*sigma^2_(t-1)

summary(garchFit(~arma(4,4)+garch(1,1), data=data.growth, trace=FALSE))

# Produces a significantly different set of coefficients if we use garchFit instead of ugarchfit.
# Nonetheless we use this to verify goodness of fit due to all of the test performed by it.

# Standardised Residuals Tests:
#                                 Statistic p-Value
```

```
# Jarque-Bera Test   R    Chi^2  55.67742  8.124612e-13
# Shapiro-Wilk Test  R    W      0.9822455 0.0001214266
# Ljung-Box Test     R    Q(10)  4.683363  0.9113019
# Ljung-Box Test     R    Q(15)  5.972257  0.98021
# Ljung-Box Test     R    Q(20)  8.220803  0.9903032
# Ljung-Box Test     R^2  Q(10)  2.410858  0.992112
# Ljung-Box Test     R^2  Q(15)  5.080218  0.9914252
# Ljung-Box Test     R^2  Q(20)  6.10085   0.9987543
# LM Arch Test       R    TR^2   4.568772  0.9708563
#
# Information Criterion Statistics:
#   AIC        BIC        SIC       HQIC
# -2.673954 -2.550014 -2.675849 -2.624784
```

Ljung-Box Test has a high p-value which means we accept the null hypothesis of uncorrelated residuals. The residual is not serially correlated.

Jarque Bera Tests has a p-value close to 0 which means we reject the null hypothesis of normality. Normality assumption of residual is violated upon this model.

ARCH Test has a high p-value so we accept the null hypothesis of constant variance. Constant variance assumption is not violated.

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

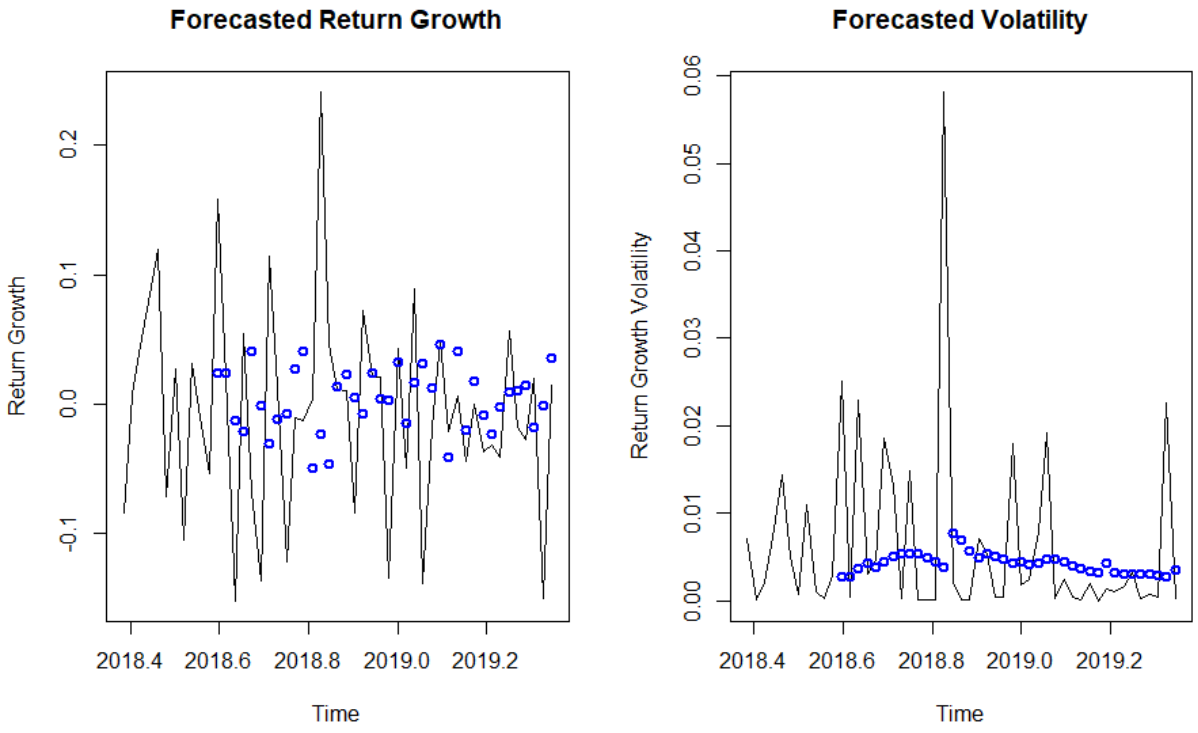
Overlay the draw an overlay of the predicted points on the original series in that range.

Forecasted values (40 time points)

```
 [1]  0.0241919571  0.0241919571 -0.0119482482 -0.0206431526  0.0406399530 -0
.0008520953 -0.0298969657
 [8] -0.0119461199 -0.0070251605  0.0275702602  0.0408178107 -0.0493253392 -0
.0233458321 -0.0463054623
[15]  0.0137737590  0.0230468753  0.0056566243 -0.0074076770  0.0239248701  0
.0043377760  0.0031536009
[22]  0.0331262798 -0.0147478513  0.0172448092  0.0313800282  0.0130861554  0
.0459488172 -0.0407311887
[29]  0.0407171968 -0.0196716251  0.0180026212 -0.0081769775 -0.0234791125 -0
.0022797031  0.0097395946
[36]  0.0109765976  0.0144790045 -0.0178810192 -0.0007656759  0.0354794175
```

Mean Absolute Percentage Error (MAPE)
mean(abs(fore.series - data.growth.test)/data.growth.test)
# 1.829894

Precision Measure (PM)
sum((fore.series - data.growth.test)^2)/sum((data.growth.test-mean(data.growth.test))^2)
# 1.136686



<Codes>

```r
# ISYE6402 HW4 npark62

library(quantmod)
library(tseries)
library(fGarch)
library(mgcv)
library(TSA)
library(rugarch)

fname <- file.choose()
data <- read.csv(fname)
data <- log(data[,2])
data.ts <- ts(data,start=c(2012,1),freq=52)
data.growth = diff(data.ts)


# Question 1

test_modelA <- function(p,d,q){
  mod = arima(data.growth, order=c(p,d,q), method="ML")
  current.aic = AIC(mod)
  df = data.frame(p,d,q,current.aic)
  names(df) <- c("p","d","q","AIC")
  print(paste(p,d,q,current.aic,sep=" "))
  return(df)
}

orders = data.frame(Inf,Inf,Inf,Inf)
names(orders) <- c("p","d","q","AIC")


for (p in 0:4){
  for (d in 0:1){
    for (q in 0:4) {
      possibleError <- tryCatch(
        orders<-rbind(orders,test_modelA(p,d,q)),
        error=function(e) e
      )
      if(inherits(possibleError, "error")) next}}}
orders <- orders[order(-orders$AIC),]
tail(orders)

# p d q       AIC
# 1 0 0  -999.979
# 3 0 4 -1000.316
# 4 0 3 -1001.778
# 0 0 0 -1001.952
# 2 0 4 -1002.094
# 4 0 2 -1003.803

# Applying no threshold, we choose ARIMA(4,0,2)

model = arima(data.growth, order=c(4,0,2), method="ML")
```

```
# Coefficients:
#          ar1      ar2     ar3     ar4     ma1     ma2  intercept
#       -1.2930  -0.8478  0.1053  0.1200  1.3200  0.9343     0.0058
# s.e.   0.0662   0.1212  0.0848  0.0573  0.0446  0.0928     0.0036
#
# sigma^2 estimated as 0.00405:  log likelihood = 509.9,  aic = -1005.8

resids = resid(model)

par(mfcol=c(1,2))
acf(resids, main = 'ACF of Residual Plot')
acf(resids^2, main = 'ACF of Squared Residual Plot')

# Both residual and squared residual ACFs somewhat resemble white noise.
# Squared residual ACF does not resemble white noise perfectly which implies
uncorrelation + dependence.


Box.test(resids, lag = 6, type = 'Ljung', fitdf = 5)
# lag number = p order + q order = 6

# Box-Ljung test
#
# data:  resids
# X-squared = 0.74572, df = 1, p-value = 0.3878

# p-value is high so we accept the null of uncorrelated residual.

Box.test(resids^2, lag = 6, type = 'Ljung', fitdf = 5)

# Box-Ljung test
#
# data:  resids^2
# X-squared = 6.6919, df = 1, p-value = 0.009685

# p-value is lower than an alpha of 0.01 so we reject the null of
uncorrelated squared residuals.


# Because squared residuals are correlated, the variance is non-constant.
(i.e. heteroskedastic)


# Question 2

#Initial GARCH Order
#ARIMA-GARCH GARCH order
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)
```

```
}

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


for (m in 0:2){
     for (n in 0:2){
          possibleError <- tryCatch(
            orders<-rbind(orders,test_modelAGG(m,n)),
            error=function(e) e
          )
          if(inherits(possibleError, "error")) next
          }
}
orders <- orders[order(-orders$BIC),]
tail(orders)

# m n       BIC
# 2 1 -2.533904
# 2 0 -2.536584
# 0 1 -2.548280
# 1 1 -2.549468
# 0 0 -2.563903
# 1 2 -2.585301

# Minimum BIC occurs when garch(m = 1, n = 2)

# First model produced, then, is ARMA(4,2)-GARCH(1,2)

#ARMA update
#ARIMA-GARCH ARIMA order
test_modelAGA <- function(p,q){
  spec = ugarchspec(variance.model=list(garchOrder=c(1,2)),
    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)
}

orders = data.frame(Inf,Inf,Inf)
names(orders) <- c("p","q","BIC")


for (p in 0:4){
     for (q in 0:4){
          possibleError <- tryCatch(
            orders<-rbind(orders,test_modelAGA(p,q)),
            error=function(e) e
          )
          if(inherits(possibleError, "error")) next
          }
}
```

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

# p q       BIC
# 0 1 -2.579060
# 1 0 -2.579097
# 4 3 -2.581914
# 4 2 -2.585301
# 0 0 -2.594143
# 4 4 -2.631685

# Lowest BIC, when GARCH(1,2), is when ARMA(p=4,q=4)
# Second model produced, then, is ARMA(4,4)-GARCH(1,2)

#GARCH update
test_modelAGG <- function(m,n){
  spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                    mean.model=list(armaOrder=c(4,4),
                      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)
}


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


for (m in 0:2){
     for (n in 0:2){
          possibleError <- tryCatch(
            orders<-rbind(orders,test_modelAGG(m,n)),
            error=function(e) e
          )
          if(inherits(possibleError, "error")) next
          }
}
orders <- orders[order(-orders$BIC),]
tail(orders)

# m n       BIC
# 0 1 -2.518347
# 2 2 -2.531000
# 0 2 -2.545794
# 1 0 -2.559581
# 1 2 -2.631685
# 1 1 -2.650102

# Lowest BIC, when ARMA(4,4), is when GARCH(1,1)
# Third produced model, then is, ARMA(4,4)-GARCH(1,1)
```

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


final.model.1 = ugarchfit(spec.1, data.growth, solver = 'hybrid')
# ARMA(4,2) - GARCH(1,2)
final.model.2 = ugarchfit(spec.2, data.growth, solver = 'hybrid')
# ARMA(4,4) - GARCH(1,2)
final.model.3 = ugarchfit(spec.3, data.growth, solver = 'hybrid')
# ARMA(4,4) - GARCH(1,1)

infocriteria(final.model.1) # BIC: -2.585301
infocriteria(final.model.2) # BIC: -2.631685
infocriteria(final.model.3) # BIC: -2.650102 ***lowest BIC

final.model = final.model.3 # ARMA(4,4) - GARCH(1,1)

# Optimal Parameters
# ------------------------------------
#   Estimate  Std. Error     t value Pr(>|t|)
# mu      0.002996    0.000031    96.12606 0.000000
# ar1    -1.580124    0.000226 -6987.71919 0.000000
# ar2    -1.565839    0.000215 -7273.18370 0.000000
# ar3    -1.561086    0.000210 -7418.86845 0.000000
# ar4    -0.961903    0.000166 -5781.61309 0.000000
# ma1     1.583024    0.000308  5140.08970 0.000000
# ma2     1.553664    0.000303  5127.27566 0.000000
# ma3     1.611279    0.000364  4420.77485 0.000000
# ma4     1.070483    0.000113  9504.65001 0.000000
# omega   0.000289    0.000125     2.31412 0.020661
# alpha1  0.055345    0.060811     0.91012 0.362757
# beta1   0.859849    0.107686     7.98474 0.000000
# shape   9.634077    3.624748     2.65786 0.007864

# Y_t = 0.002996  - 1.580124*Y_(t-1) -1.565839*Y_(t-2) -1.561086*Y_(t-3) -
0.961903*Y_(t-4) + Z_t + 1.583024*Z_(t-1)+1.553664*Z_(t-2) +1.611279*Z_(t-3)
+ 1.070483*Z_(t-4)
# Sigma^2_t = 0.000289 + 0.055345*Z^2_(t-1) + 0.859849*sigma^2_(t-1)

summary(garchFit(~arma(4,4)+garch(1,1), data=data.growth, trace=FALSE))
# Produces a significantly different set of coefficients if we use garchFit
instead of ugarchfit.
# Nonetheless we use this to verify goodness of fit due to all of the test
performed by it.

# Standardised Residuals Tests:
#                                 Statistic p-Value
# Jarque-Bera Test   R    Chi^2  55.67742  8.124612e-13
# Shapiro-Wilk Test  R    W      0.9822455 0.0001214266
# Ljung-Box Test     R    Q(10)  4.683363  0.9113019
```

```
# Ljung-Box Test     R    Q(15)  5.972257  0.98021
# Ljung-Box Test     R    Q(20)  8.220803  0.9903032
# Ljung-Box Test     R^2  Q(10)  2.410858  0.992112
# Ljung-Box Test     R^2  Q(15)  5.080218  0.9914252
# Ljung-Box Test     R^2  Q(20)  6.10085   0.9987543
# LM Arch Test       R    TR^2   4.568772  0.9708563
#
# Information Criterion Statistics:
#   AIC       BIC       SIC      HQIC
# -2.673954 -2.550014 -2.675849 -2.624784

# Ljung-Box Test has a high p-value which means we accept the null hypothesis
of uncorrelated residuals. The residual is not serially correlated.
# Jarque Bera Tests has a p-value close to 0 which means we reject the null
hypothesis of normality. Normality assumption of residual is violated upon
this model.
# ARCH Test has a high p-value so we accept the null hypothesis of constant
variance. Constant variance assumption is not violated.

# Question 3

n = length(data.growth)
data.growth.train = data.growth[1:(n-40)]
data.growth.test = data.growth[(n-39):n]
nfore = length(data.growth.test)
vol = time(data.growth)

fore.series = NULL
fore.sigma = NULL

for(f in 1:nfore){
  data = data.growth.train
  if (f>2)
      data = c(data.growth.train,data.growth.test[1:(f-1)])
  final.model = ugarchfit(spec.3, data, solver = 'hybrid')
  fore = ugarchforecast(final.model, n.ahead = 1)
  fore.series = c(fore.series,fore@forecast$seriesFor)
  fore.sigma = c(fore.sigma, fore@forecast$sigmaFor)
}


### Mean Absolute Percentage Error (MAPE)
mean(abs(fore.series - data.growth.test)/data.growth.test)
# 1.829894

### Precision Measure (PM)
sum((fore.series - data.growth.test)^2)/sum((data.growth.test-
mean(data.growth.test))^2)
# 1.136686


ymin = min(c(as.vector(data.growth.test),fore.series))
ymax = max(c(as.vector(data.growth.test),fore.series))

plot(vol[(n-50):n],data.growth[(n-50):n],type="l",ylim=c(ymin,ymax), xlab=
"Time",ylab= "Return Growth",main= "Forecasted Return Growth")
points(vol[(n-39):n],fore.series, lwd=2 , col="blue")
```

```
ymin = min(c(as.vector(data.growth.test^2),fore.sigma^2))
ymax = max(c(as.vector(data.growth.test^2),fore.sigma^2))

plot(vol[(n-50):n],data.growth[(n-50):n]^2,type="l", ylim=c(ymin,ymax),
xlab="Time", ylab="Return Growth Volatility",main= "Forecasted Volatility")
points(vol[(n-39):n],fore.sigma^2,lwd= 2, col="blue")
```