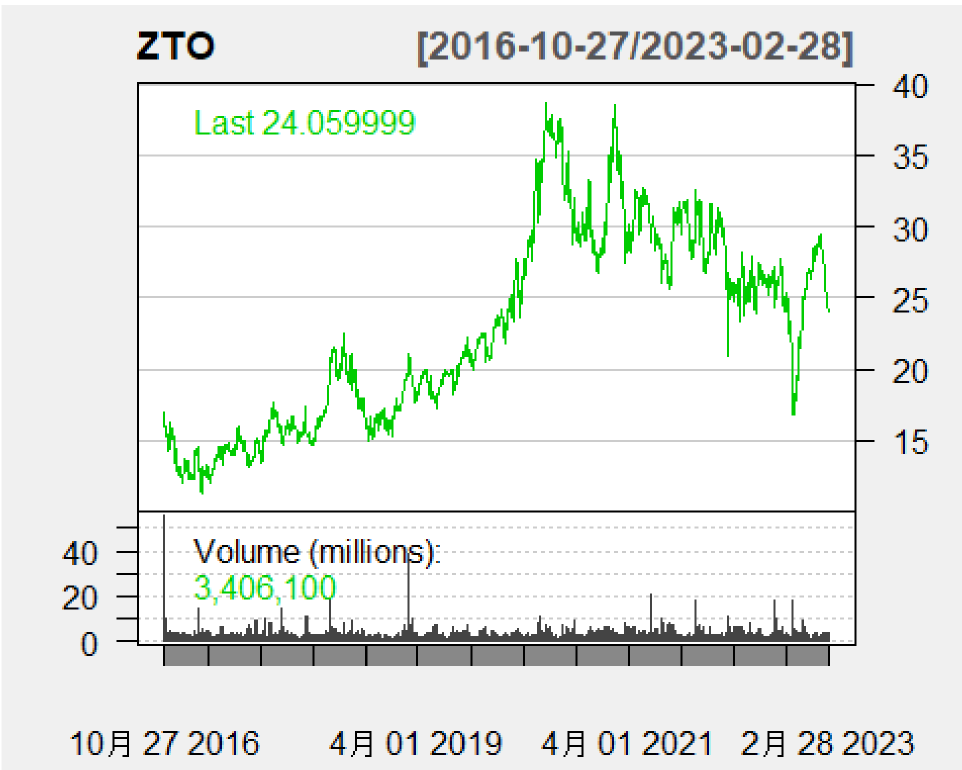
Appendix

1. Data prepare

```
# Load packages
library(quantmod)

## 载入需要的程辑包：xts

## Warning: 程辑包'xts'是用R版本4.2.3 来建造的

## 载入需要的程辑包：zoo

## Warning: 程辑包'zoo'是用R版本4.2.3 来建造的

##
## 载入程辑包：'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_x
ts) #
## # that you enter or source() into this session won't work correctly.
   #
## #
   #
## # All package code is unaffected because it is protected by the R namespac
e   #
## # mechanism.
   #
## #
   #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warnin
g.  #
## #
   #
## # 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 sto
```

```
p   #
## # dplyr from breaking base R's lag() function.
    #
## ################################### WARNING ##############################
#####

## 载入需要的程辑包：TTR

## Warning: 程辑包'TTR'是用R版本4.2.3 来建造的

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

# Load QuantMod package
library(quantmod)

# Set the ticker symbol
ticker <- "ZTO"

# Set the start and end dates for the data
start_date <- as.Date("2011-01-01")
end_date <- as.Date("2023-03-01")

# Download the data
getSymbols(ticker, src = "yahoo", from = start_date, to = end_date)

## [1] "ZTO"

# View the data
head(ZTO)

##            ZTO.Open ZTO.High ZTO.Low ZTO.Close ZTO.Volume ZTO.Adjusted
## 2016-10-27    18.40    18.45  16.500     16.57   55321100     15.46712
## 2016-10-28    16.79    17.25  16.680     16.99   12646800     15.85917
## 2016-10-31    17.19    17.19  16.850     16.93    3226600     15.80316
## 2016-11-01    17.00    17.05  15.545     16.00    9863300     14.93506
## 2016-11-02    16.33    16.39  15.820     16.00    4429500     14.93506
## 2016-11-03    16.05    16.05  15.780     15.99    2318100     14.92573

# View data
str(ZTO)

## An xts object on 2016-10-27 / 2023-02-28 containing:
##   Data:    double [1594, 6]
##   Columns: ZTO.Open, ZTO.High, ZTO.Low, ZTO.Close, ZTO.Volume ... with 1 m
ore column
##   Index:   Date [1594] (TZ: "UTC")
##   xts Attributes:
##     $ src    : chr "yahoo"
##     $ updated: POSIXct[1:1], format: "2023-04-06 17:05:19"
```

```r
# extract columns, we use Op, Hi, Lo, Cl, Vo and Ad
Open <- Op(ZTO)   #Open Price
High <- Hi(ZTO)    # High price
Low <- Lo(ZTO)  # Low price
Close<- Cl(ZTO)   #Close Price
Volume <- Vo(ZTO)   #Volume
AdjClose <- Ad(ZTO) # Adjusted close

# Plot the chart
chartSeries(ZTO,
            type="line",
            theme=chartTheme('white'))
```

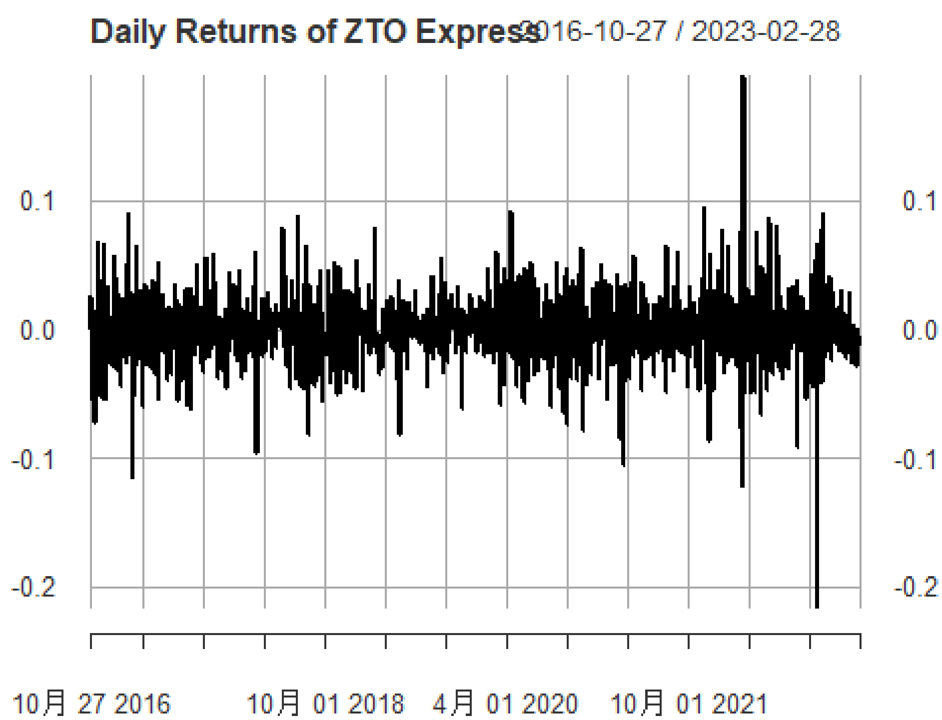


green color means that the return is positive

```r
library(TTR)

# Simple Moving Average
sma <-SMA(Cl(ZTO),n=20)
tail(sma,n=5)

##                 SMA
## 2023-02-22 27.6315
## 2023-02-23 27.4220
## 2023-02-24 27.1825
## 2023-02-27 26.9370
## 2023-02-28 26.7100
```

2. Modeling

```
# Calculate the daily returns
library(quantmod)
returns <- dailyReturn(Cl(ZTO))

# Plot the returns
plot(returns, main = "Daily Returns of ZTO Express")
```



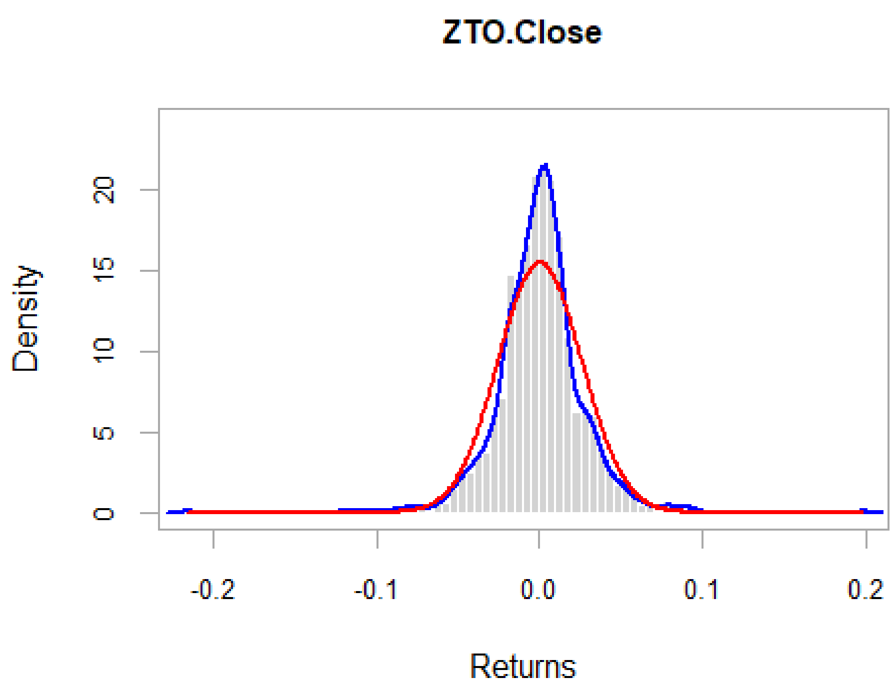

```
tail(returns)

##            daily.returns
## 2023-02-21 -0.0035985606
## 2023-02-22  0.0004012841
## 2023-02-23 -0.0076213398
## 2023-02-24 -0.0117218674
## 2023-02-27 -0.0102249485
## 2023-02-28 -0.0057852064
```

stationality transformation

```
library(PerformanceAnalytics)

## Warning: 程辑包'PerformanceAnalytics'是用R版本4.2.3 来建造的

##
## 载入程辑包：'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```r
# Calculate the returns
returns <- CalculateReturns(Cl(ZTO))

# Remove any missing values
returns <- na.omit(returns)

# Create a histogram with a normal curve overlayx
chart.Histogram(returns,
                methods = c('add.density', 'add.normal'),
                colorset = c('light gray', 'blue', 'red'))
```




```r
library(tseries)
# verify the stationarity of the returns using Augmented Dickey-Fuller test
adf.test(returns)

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

##
##  Augmented Dickey-Fuller Test
##
## data:  returns
## Dickey-Fuller = -12.581, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

When the p-value is small (typically less than 0.01), it is generally taken as strong evidence to reject the null hypothesis. In the context of time series analysis, this often indicates that the time series is stationary. Specifically, a small p-value suggests that the data deviates significantly from what would be expected under the null hypothesis of non-stationarity. Therefore, we can conclude that the time series exhibits some form of stationarity, such as weak or strong stationarity.

```
# Split the data into training and validation sets
library(caret)

## 载入需要的程辑包：ggplot2

## 载入需要的程辑包：lattice

set.seed(123)
# Set the train start and end dates
train_start_date <- as.Date("2016-01-01")
train_end_date <- as.Date("2022-12-31")

# Set the test start and end dates
test_start_date <- as.Date("2023-01-01")
test_end_date <- as.Date("2023-03-01")

# Filter the returns data for the train and test sets
train_returns <- returns[train_start_date <= index(returns) & index(returns)
<= train_end_date]
test_returns <- returns[test_start_date <= index(returns) & index(returns) <=
 test_end_date]
```

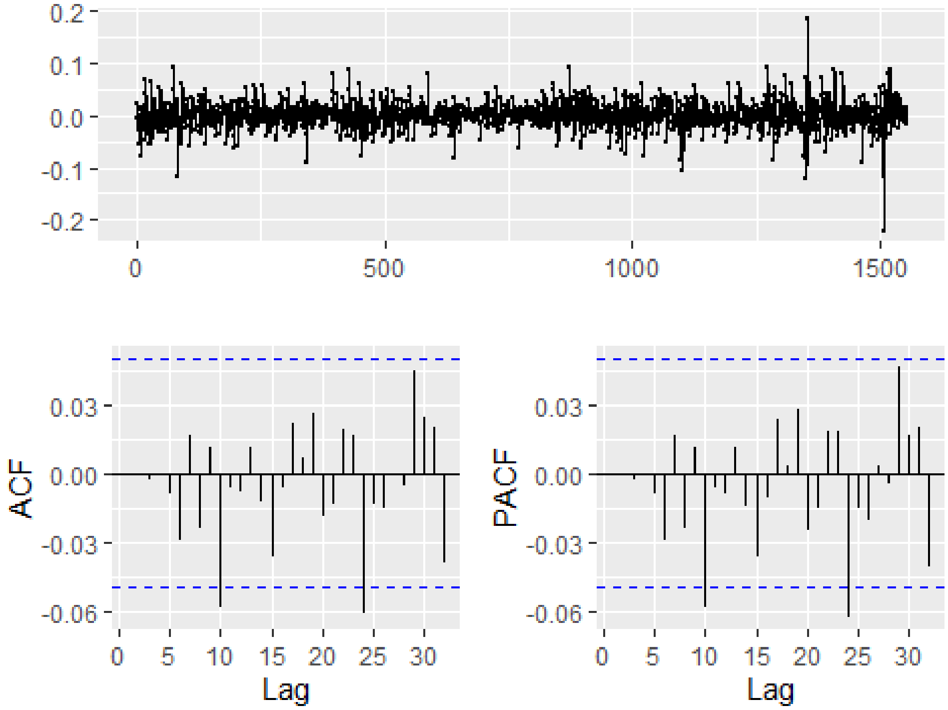
ARIMA models

```
# Fitting ARIMA
library(forecast)

## Warning: 程辑包'forecast'是用 R 版本 4.2.3 来建造的

arima_model = auto.arima(train_returns , max.order = c(3 , 0 ,3) , stationary
 = TRUE , trace = T , ic = 'aic')

##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : -6937.124
##  ARIMA(0,0,0) with non-zero mean : -6931.472
##  ARIMA(1,0,0) with non-zero mean : -6929.507
##  ARIMA(0,0,1) with non-zero mean : -6929.613
##  ARIMA(0,0,0) with zero mean     : -6932.498
##  ARIMA(1,0,2) with non-zero mean : -6930.859
##  ARIMA(2,0,1) with non-zero mean : -6936.449
##  ARIMA(3,0,2) with non-zero mean : -6936.081
##  ARIMA(2,0,3) with non-zero mean : -6933.189
```

```
##  ARIMA(1,0,1) with non-zero mean : -6927.51
##  ARIMA(1,0,3) with non-zero mean : -6935.876
##  ARIMA(3,0,1) with non-zero mean : -6937.893
##  ARIMA(3,0,0) with non-zero mean : -6939.892
##  ARIMA(2,0,0) with non-zero mean : -6928.548
##  ARIMA(4,0,0) with non-zero mean : -6936.892
##  ARIMA(4,0,1) with non-zero mean : Inf
##  ARIMA(3,0,0) with zero mean     : -6940.571
##  ARIMA(2,0,0) with zero mean     : -6929.527
##  ARIMA(4,0,0) with zero mean     : -6937.571
##  ARIMA(3,0,1) with zero mean     : -6938.57
##  ARIMA(2,0,1) with zero mean     : -6934.785
##  ARIMA(4,0,1) with zero mean     : -6935.571
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,0,0) with zero mean     : -6938.24
##
##  Best model: ARIMA(3,0,0) with zero mean
```

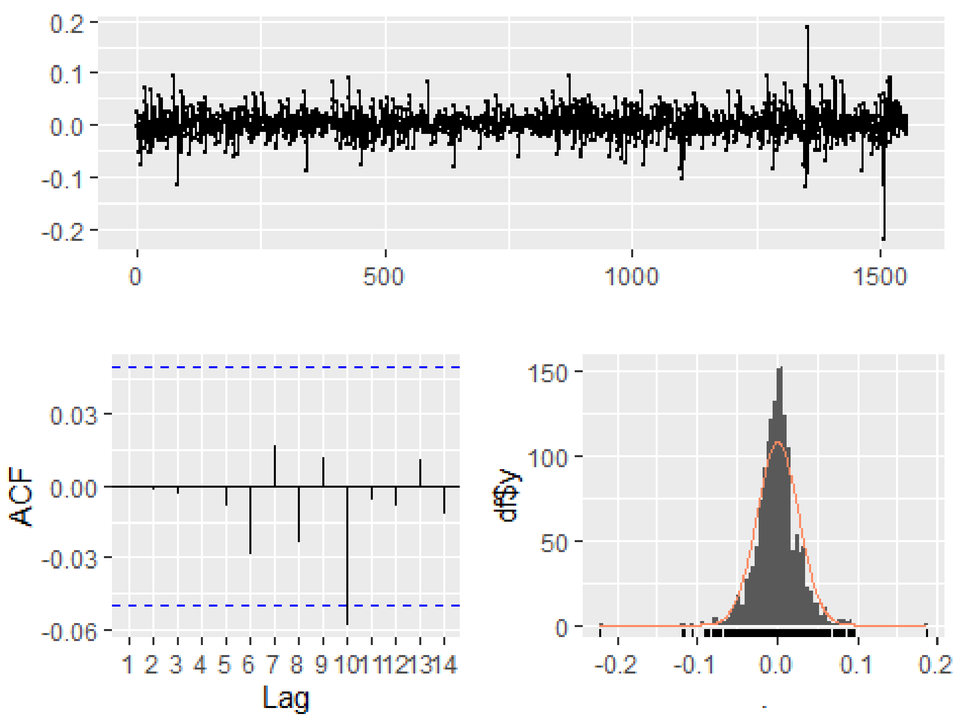
the minimum AIC = -6938.24 Therefore, ARIMA(3,0,0) is the best. 3-order Autoregressive (AR(3)).

```
# Check estimation
arima_model

## Series: train_returns
## ARIMA(3,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2      ar3
##       -0.0115  -0.0361  -0.0789
## s.e.   0.0253   0.0253   0.0253
##
## sigma^2 = 0.0006716:  log likelihood = 3473.12
## AIC=-6938.24   AICc=-6938.21   BIC=-6916.85

# Diagnostics checking
fit <- Arima(train_returns,order =c(3,0,0))
ggtsdisplay(resid(fit))
```



residuals seem to be white noise

```
arima_model$residuals %>% ggtsdisplay(plot.type = 'hist' , lag.max = 14)
```



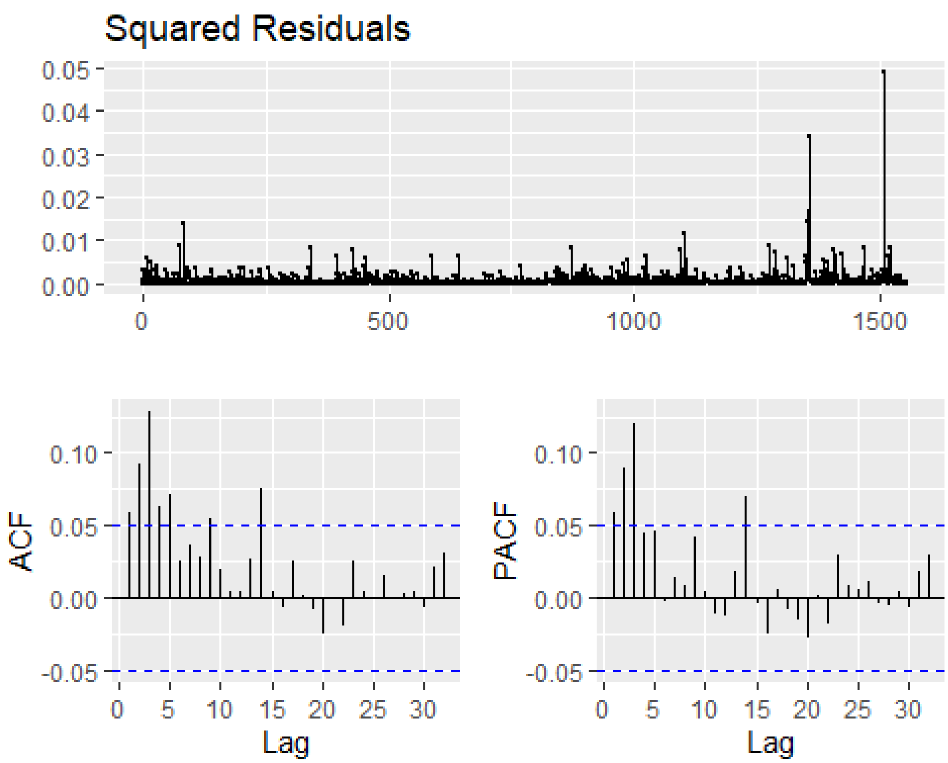
```r
# Test if the residuals are white noise
returns_ar = arima_model$residuals
Box.test(arima_model$residuals , lag = 14 , fitdf = 2 , type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  arima_model$residuals
## X-squared = 8.8482, df = 12, p-value = 0.7158
```

It is white noise.

GARCH

```r
# Squares of the residuals
ggtsdisplay(resid(fit)^2, main = "Squared Residuals")
```



The GARCH process is valid when the squared residuals are correlated. ACF and PACF plots indicate significant correlation clearly.

We need to see whether the residuals are independent or not. ACF and PACF indicate that we need to use GARCH(p,q) they typically show significant autocorrelation and partial autocorrelation at the first few lags, followed by a sharp cutoff or decay.

```r
library(rugarch)
```

```
## Warning: 程辑包'rugarch'是用 R 版本 4.2.3 来建造的

## 载入需要的程辑包：parallel
```

```
##
## 载入程辑包：'rugarch'

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

```
# Model ARIMA(3,0,0) + GARCH(1,1)
model_spec1 = ugarchspec(variance.model = list(model = 'sGARCH' ,
                                               garchOrder = c(1 , 1)) ,
                         mean.model = list(armaOrder = c(3 , 0)))

model_fit1 = ugarchfit(spec = model_spec1 , data = returns_ar , solver = 'sol
np')

print(model_fit1)
```

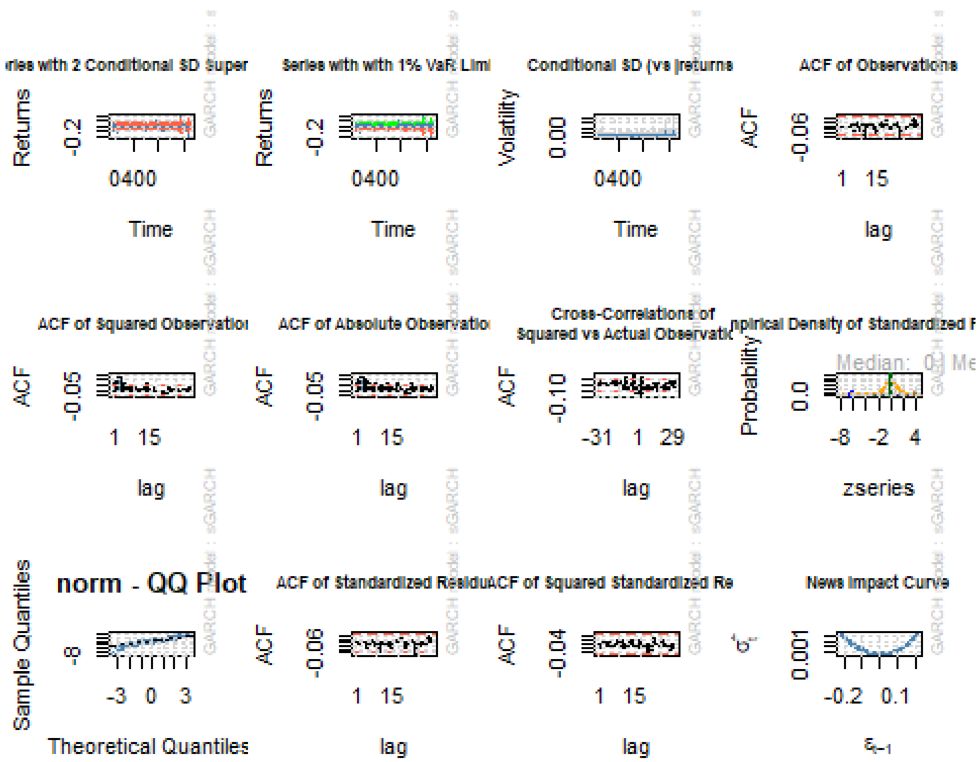
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model   : sGARCH(1,1)
## Mean Model    : ARFIMA(3,0,0)
## Distribution  : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001087    0.000628  1.730110 0.083611
## ar1    -0.001758    0.028228 -0.062265 0.950351
## ar2     0.025570    0.027900  0.916467 0.359422
## ar3     0.016984    0.027705  0.613025 0.539860
## omega   0.000036    0.000010  3.430786 0.000602
## alpha1  0.074998    0.014909  5.030358 0.000000
## beta1   0.873407    0.024036 36.337361 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001087    0.000635  1.710051 0.087256
## ar1    -0.001758    0.027708 -0.063436 0.949420
## ar2     0.025570    0.027112  0.943104 0.345628
## ar3     0.016984    0.029220  0.581233 0.561083
## omega   0.000036    0.000016  2.244698 0.024788
## alpha1  0.074998    0.030133  2.488924 0.012813
## beta1   0.873407    0.038858 22.476885 0.000000
##
## LogLikelihood : 3528.522
```

```
##
## Information Criteria
## ------------------------------------
##
## Akaike       -4.5322
## Bayes        -4.5081
## Shibata      -4.5322
## Hannan-Quinn -4.5232
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1166  0.7328
## Lag[2*(p+q)+(p+q)-1][8]     1.1132  1.0000
## Lag[4*(p+q)+(p+q)-1][14]    4.0057  0.9740
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4589  0.4981
## Lag[2*(p+q)+(p+q)-1][5]    0.5881  0.9433
## Lag[4*(p+q)+(p+q)-1][9]    1.0335  0.9848
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.08594 0.500 2.000  0.7694
## ARCH Lag[5]   0.24460 1.440 1.667  0.9546
## ARCH Lag[7]   0.45769 2.315 1.543  0.9821
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1418
## Individual Statistics:
## mu     0.04180
## ar1    0.05487
## ar2    0.20061
## ar3    0.12714
## omega  0.39871
## alpha1 0.29120
## beta1  0.29439
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
```

```
## ------------------------------------
##                       t-value   prob sig
## Sign Bias              0.4687 0.6394
## Negative Sign Bias     0.4853 0.6276
## Positive Sign Bias     0.3238 0.7462
## Joint Effect           0.6429 0.8865
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     84.30    3.336e-10
## 2    30     97.54    2.408e-09
## 3    40    108.24    1.978e-08
## 4    50    121.10    4.895e-08
##
##
## Elapsed time : 0.419461
```

```
plot(model_fit1, which = "all")
```

```
##
## please wait...calculating quantiles...
```




```
# Forecast the model
forest_garch1 <- ugarchforecast(fitORspec = model_fit1, n.ahead = 100)
```

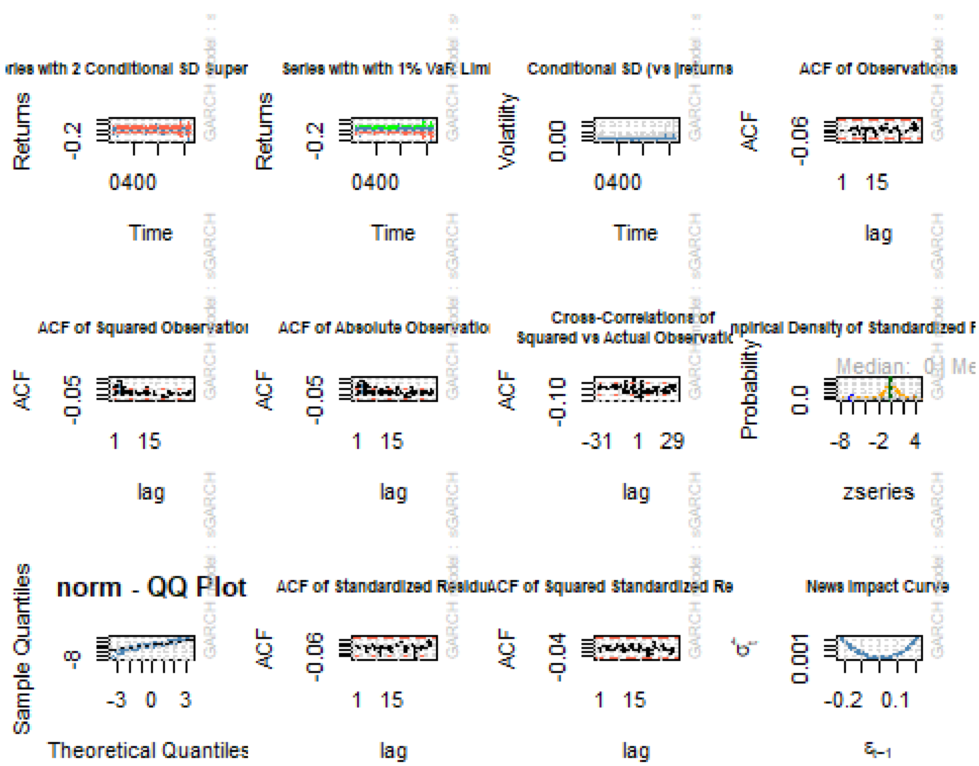
```r
# For comparison, we make the forecasts as vector
vector_f_garch1 <- as.vector(forest_garch1@forecast$seriesFor)
vector_f_garch1_sigma <- as.vector(forest_garch1@forecast$sigmaFor)

accuracy(vector_f_garch1, test_returns )

##                     ME       RMSE        MAE      MPE     MAPE
## Test set -0.003820514 0.01492682 0.01184839 92.25061 101.0076

# Model ARIMA(3,0,0) + GARCH(1,2)
model_spec2 = ugarchspec(variance.model = list(model = 'sGARCH' ,
                                               garchOrder = c(1 , 2)),
                         mean.model = list(armaOrder = c(3 , 0)))

model_fit2 = ugarchfit(spec = model_spec2 , data = returns_ar , solver = 'sol
np')

print(model_fit2)

## 
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
## 
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model   : sGARCH(1,2)
## Mean Model    : ARFIMA(3,0,0)
## Distribution  : norm
## 
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001086    0.000628  1.729876 0.083653
## ar1    -0.001767    0.028240 -0.062566 0.950112
## ar2     0.025578    0.028189  0.907380 0.364206
## ar3     0.016987    0.027794  0.611171 0.541086
## omega   0.000036    0.000017  2.063355 0.039079
## alpha1  0.074949    0.035774  2.095066 0.036165
## beta1   0.873497    0.586113  1.490320 0.136140
## beta2   0.000000    0.533250  0.000001 0.999999
## 
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001086    0.000635  1.711391 0.087009
## ar1    -0.001767    0.027587 -0.064045 0.948934
## ar2     0.025578    0.030324  0.843480 0.398960
## ar3     0.016987    0.029574  0.574395 0.565700
## omega   0.000036    0.000035  1.008930 0.313008
## alpha1  0.074949    0.065464  1.144898 0.252252
## beta1   0.873497    1.383202  0.631504 0.527711
```

```
## beta2   0.000000    1.270579  0.000000 1.000000
##
## LogLikelihood : 3528.522
##
## Information Criteria
## ------------------------------------
##
## Akaike       -4.5309
## Bayes        -4.5034
## Shibata      -4.5310
## Hannan-Quinn -4.5207
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1167  0.7326
## Lag[2*(p+q)+(p+q)-1][8]    1.1134  1.0000
## Lag[4*(p+q)+(p+q)-1][14]   4.0056  0.9740
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4580  0.4986
## Lag[2*(p+q)+(p+q)-1][8]    0.8976  0.9806
## Lag[4*(p+q)+(p+q)-1][14]   1.9661  0.9918
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]   0.08447 0.500 2.000  0.7713
## ARCH Lag[6]   0.41189 1.461 1.711  0.9162
## ARCH Lag[8]   0.89485 2.368 1.583  0.9392
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.2841
## Individual Statistics:
## mu     0.04176
## ar1    0.05483
## ar2    0.20068
## ar3    0.12710
## omega  0.39702
## alpha1 0.29004
## beta1  0.29321
## beta2  0.28785
##
## Asymptotic Critical Values (10% 5% 1%)
```

```
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.4687 0.6393
## Negative Sign Bias  0.4848 0.6279
## Positive Sign Bias  0.3233 0.7465
## Joint Effect        0.6422 0.8867
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     84.30    3.336e-10
## 2    30     96.15    3.996e-09
## 3    40    108.24    1.978e-08
## 4    50    121.10    4.895e-08
##
##
## Elapsed time : 0.4392211
```

```
plot(model_fit2, which = "all")
```

```
##
## please wait...calculating quantiles...
```



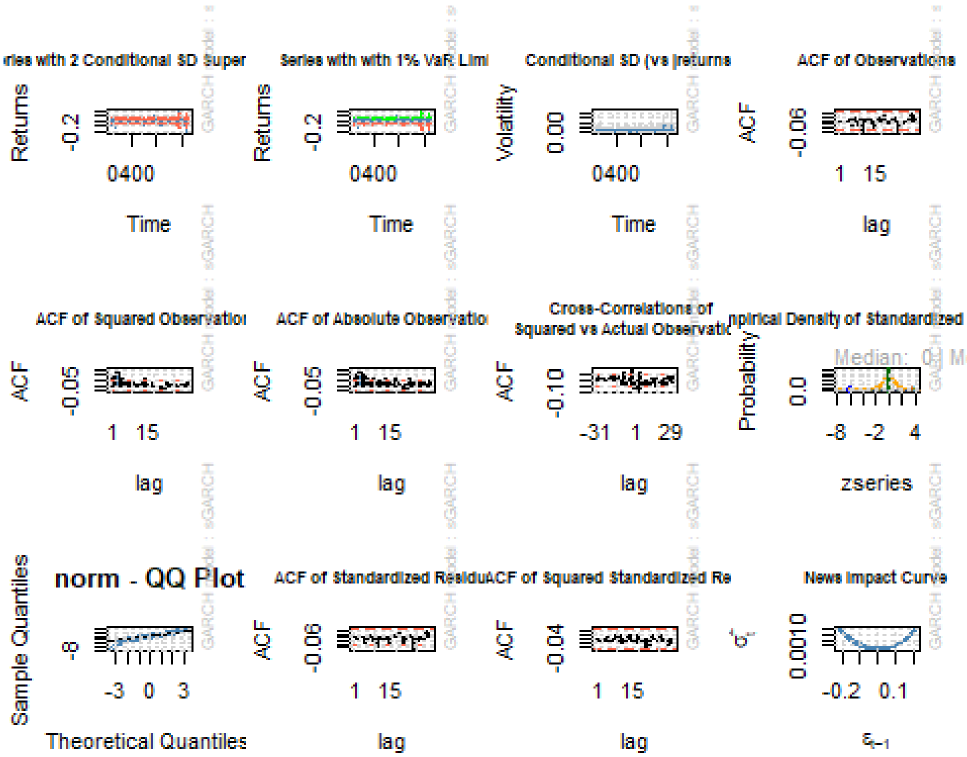
```r
# Forecast the model
forest_garch2 <- ugarchforecast(fitORspec = model_fit2, n.ahead = 100)

# For comparison, we make the forecasts as vector
vector_f_garch2 <- as.vector(forest_garch2@forecast$seriesFor)
vector_f_garch2_sigma <- as.vector(forest_garch2@forecast$sigmaFor)

accuracy(vector_f_garch2, test_returns )

##                     ME       RMSE        MAE      MPE     MAPE
## Test set -0.003820387 0.01492678 0.01184837 92.25153 101.0069

# Model ARMIA(3,0,0) + GARCH(2,1)
model_spec3 = ugarchspec(variance.model = list(model = 'sGARCH' ,
                                               garchOrder = c(2 , 1)) ,
                         mean.model = list(armaOrder = c(3 , 0)))

model_fit3 = ugarchfit(spec = model_spec3 , data = returns_ar , solver = 'sol
np')

print(model_fit3)

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(3,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001076    0.000628  1.712534 0.086798
## ar1    -0.001043    0.026997 -0.038622 0.969191
## ar2     0.028737    0.028207  1.018785 0.308305
## ar3     0.014689    0.027984  0.524900 0.599653
## omega   0.000043    0.000013  3.297324 0.000976
## alpha1  0.039125    0.026216  1.492390 0.135597
## alpha2  0.045902    0.031000  1.480714 0.138683
## beta1   0.853032    0.030070 28.368133 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001076    0.000626  1.719278 0.085564
## ar1    -0.001043    0.027116 -0.038454 0.969326
## ar2     0.028737    0.028094  1.022898 0.306356
```

```
## ar3     0.014689    0.028695  0.511889 0.608728
## omega   0.000043    0.000020  2.122614 0.033786
## alpha1  0.039125    0.035743  1.094618 0.273684
## alpha2  0.045902    0.051718  0.887540 0.374788
## beta1   0.853032    0.050816 16.786735 0.000000
##
## LogLikelihood : 3529.496
##
## Information Criteria
## ------------------------------------
##
## Akaike       -4.5322
## Bayes        -4.5046
## Shibata      -4.5322
## Hannan-Quinn -4.5219
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1129  0.7369
## Lag[2*(p+q)+(p+q)-1][8]     1.1595  1.0000
## Lag[4*(p+q)+(p+q)-1][14]    4.0518  0.9719
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.01181  0.9135
## Lag[2*(p+q)+(p+q)-1][8]    0.50636  0.9961
## Lag[4*(p+q)+(p+q)-1][14]   1.62175  0.9965
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]   0.09167 0.500 2.000  0.7621
## ARCH Lag[6]   0.46187 1.461 1.711  0.9026
## ARCH Lag[8]   0.96756 2.368 1.583  0.9290
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1481
## Individual Statistics:
## mu     0.03581
## ar1    0.04893
## ar2    0.21088
## ar3    0.10304
## omega  0.38851
## alpha1 0.30150
```

```
## alpha2 0.30394
## beta1  0.29716
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.42625 0.6700
## Negative Sign Bias 0.02978 0.9762
## Positive Sign Bias 0.04826 0.9615
## Joint Effect       0.30594 0.9589
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     80.03    1.839e-09
## 2    30     92.10    1.723e-08
## 3    40    111.38    6.890e-09
## 4    50    131.59    1.738e-09
##
##
## Elapsed time : 0.4001622
```

```
plot(model_fit3, which = "all")
```

```
##
## please wait...calculating quantiles...
```

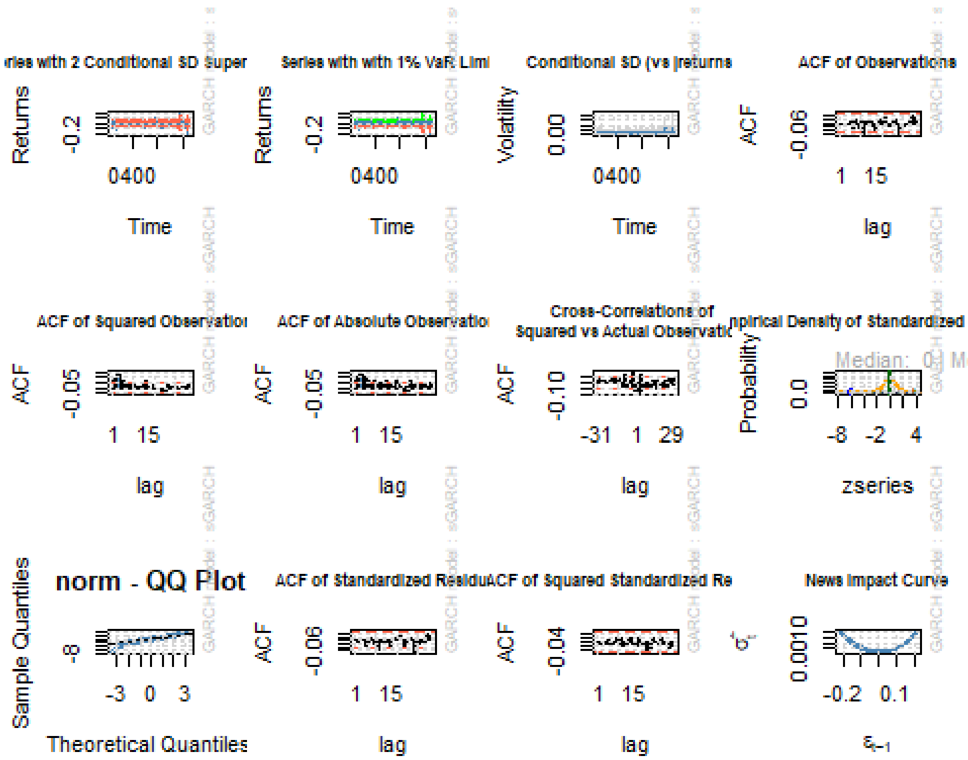



```r
# Forecast the model
forest_garch3 <- ugarchforecast(fitORspec = model_fit3, n.ahead = 100)

# For comparison, we make the forecasts as vector
vector_f_garch3 <- as.vector(forest_garch3@forecast$seriesFor)
vector_f_garch3_sigma <- as.vector(forest_garch3@forecast$sigmaFor)

accuracy(vector_f_garch3, test_returns )

##                       ME       RMSE        MAE      MPE    MAPE
## Test set -0.003811172 0.01492215 0.01184575 92.32297 100.941

# Model ARIMA(3,0,0) + GARCH(2,2)
model_spec4 = ugarchspec(variance.model = list(model = 'sGARCH' ,
                                               garchOrder = c(2 , 2)) ,
                         mean.model = list(armaOrder = c(3 , 0)))

model_fit4 = ugarchfit(spec = model_spec4 , data = returns_ar , solver = 'sol
np')

print(model_fit4)

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
```

```
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,2)
## Mean Model   : ARFIMA(3,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001076    0.000629  1.709671 0.087327
## ar1    -0.001034    0.027248 -0.037944 0.969733
## ar2     0.028700    0.028234  1.016525 0.309380
## ar3     0.014686    0.028339  0.518215 0.604308
## omega   0.000043    0.000022  1.963052 0.049640
## alpha1  0.039135    0.026689  1.466324 0.142560
## alpha2  0.045889    0.046053  0.996452 0.319030
## beta1   0.853110    0.549592  1.552261 0.120600
## beta2   0.000003    0.484558  0.000006 0.999995
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001076    0.000627  1.714548 0.086428
## ar1    -0.001034    0.027845 -0.037130 0.970381
## ar2     0.028700    0.028306  1.013934 0.310614
## ar3     0.014686    0.029771  0.493294 0.621805
## omega   0.000043    0.000029  1.487527 0.136876
## alpha1  0.039135    0.039410  0.993025 0.320698
## alpha2  0.045889    0.044749  1.025489 0.305133
## beta1   0.853110    0.737863  1.156189 0.247604
## beta2   0.000003    0.661747  0.000005 0.999996
##
## LogLikelihood : 3529.496
##
## Information Criteria
## ------------------------------------
##
## Akaike       -4.5309
## Bayes        -4.4999
## Shibata      -4.5309
## Hannan-Quinn -4.5194
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1127  0.7371
## Lag[2*(p+q)+(p+q)-1][8]     1.1581  1.0000
## Lag[4*(p+q)+(p+q)-1][14]    4.0502  0.9720
## d.o.f=3
## H0 : No serial correlation
##
```

```
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.0118  0.9135
## Lag[2*(p+q)+(p+q)-1][11]    0.9918  0.9966
## Lag[4*(p+q)+(p+q)-1][19]    2.6740  0.9974
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.1020 0.500 2.000  0.7495
## ARCH Lag[7]    0.5464 1.473 1.746  0.8846
## ARCH Lag[9]    1.1749 2.402 1.619  0.9060
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1725
## Individual Statistics:
## mu     0.03580
## ar1    0.04895
## ar2    0.21040
## ar3    0.10308
## omega  0.38783
## alpha1 0.30145
## alpha2 0.30395
## beta1  0.29688
## beta2  0.28710
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.42623 0.6700
## Negative Sign Bias 0.03006 0.9760
## Positive Sign Bias 0.04794 0.9618
## Joint Effect       0.30591 0.9589
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     79.23    2.523e-09
## 2    30     92.10    1.723e-08
## 3    40    110.81    8.341e-09
## 4    50    130.62    2.378e-09
##
```

```
##
## Elapsed time : 0.662308

plot(model_fit4, which = "all")

##
## please wait...calculating quantiles...
```




```
# Forecast the model
forest_garch4 <- ugarchforecast(fitORspec = model_fit4, n.ahead = 100)

# For comparison, we make the forecasts as vector
vector_f_garch4 <- as.vector(forest_garch4@forecast$seriesFor)
vector_f_garch4_sigma <- as.vector(forest_garch4@forecast$sigmaFor)

accuracy(vector_f_garch4, test_returns )

##                      ME       RMSE        MAE      MPE     MAPE
## Test set -0.003811163 0.01492218 0.01184577 92.32297 100.9411
```

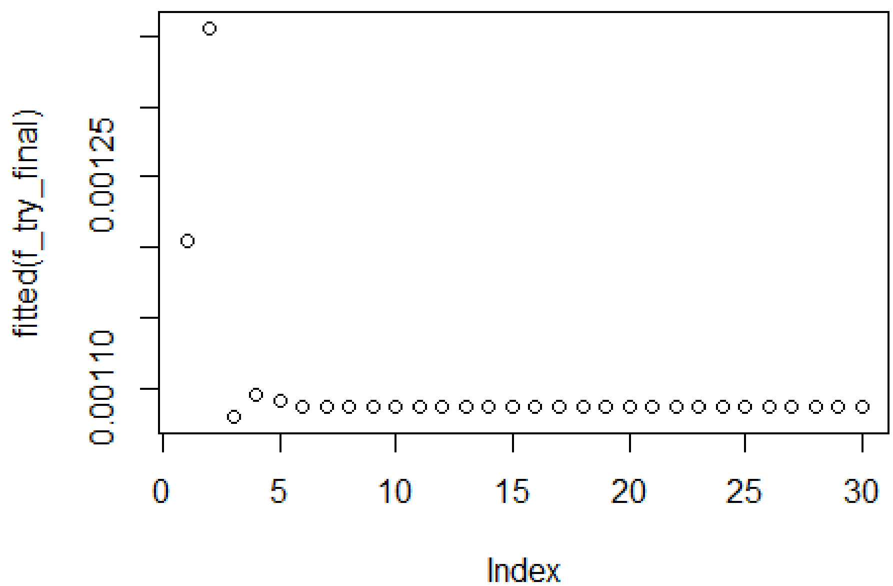
Since the value of ME, RMSE, MAE, MPE, and MAPE's value for each model are too similar, therefore, I use AIC, BIC, HQIC to find which one is the best model.

By comparing the AIC, BIC, HQIC, find that ARIMA(3,0,0) + GARCH(1,1) is the best one.

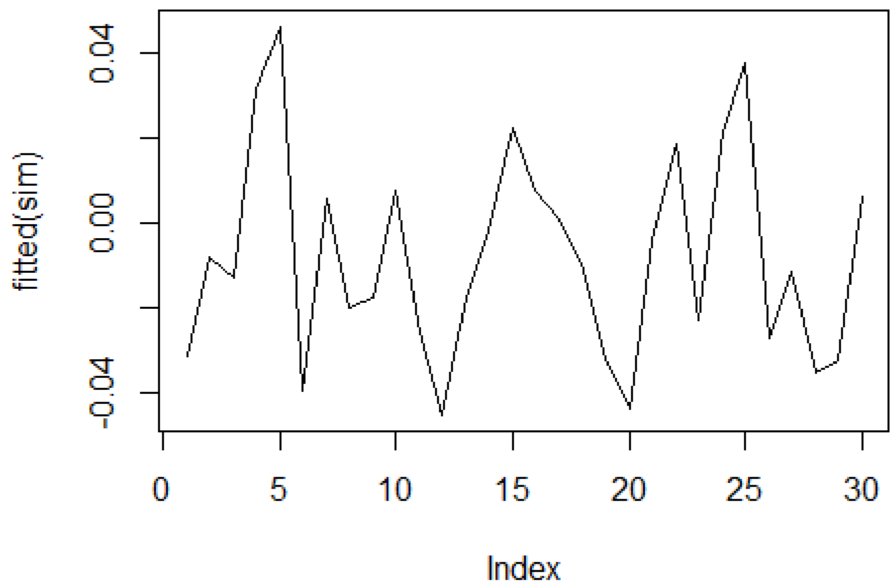
Forecast

```
# Model ARIMA(3,0,0) + GARCH(1,1)
model_spec1_final = ugarchspec(variance.model = list(model = 'sGARCH' ,
```

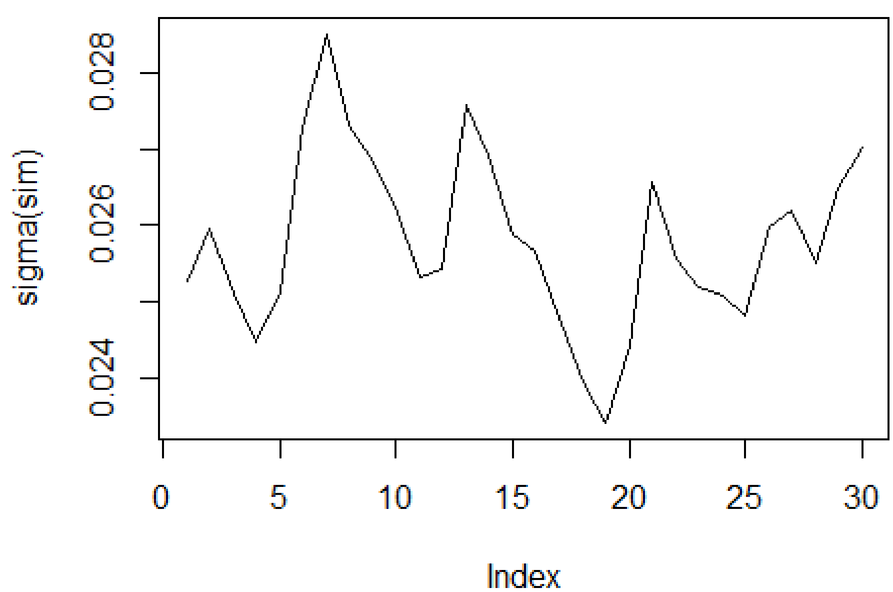
```
                                           garchOrder = c(1 , 1)) ,
                          mean.model = list(armaOrder = c(3 , 0)))

model_fit1_final = ugarchfit(spec = model_spec1_final , data = returns_ar , s
olver = 'solnp')
f_try_final <- ugarchforecast(fitORspec = model_fit1_final, n.ahead = 30)
plot(fitted(f_try_final))
```




```
final <- model_spec1_final
setfixed(final) <- as.list(coef(model_fit1_final))

sim <- ugarchpath(spec = final,
                  m.sim = 1,
                  n.sim = 1*30,
                  rseed = 30)
plot.zoo(fitted(sim))
```




```
plot.zoo(sigma(sim))
```



```r
prediction_ZTO<- 24.06*apply(fitted(sim), 2, 'cumsum') + 24.06
matplot(prediction_ZTO, type = "l", lwd = 3)
```