Cardano.R

Shankii

2023-04-24

```
library(tseries)
```

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

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(stats)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
##
## Attaching package: 'TSA'
##
## The following object is masked from 'package:readr':
##
##     spec
##
## The following objects are masked from 'package:stats':
##
##     acf, arima
##
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(urca)
```

```
## Warning: package 'urca' was built under R version 4.2.3
```

```
library(FinTS)
```

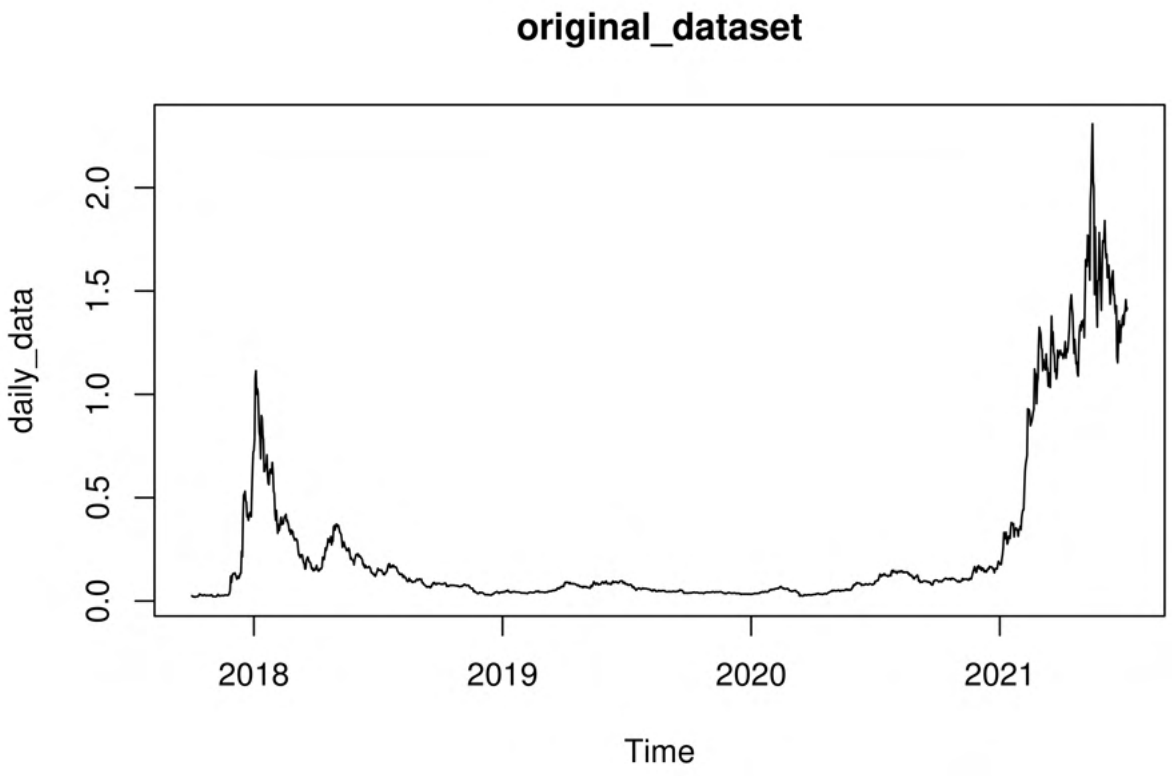
```
## Warning: package 'FinTS' was built under R version 4.2.3
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## Attaching package: 'FinTS'
##
## The following object is masked from 'package:forecast':
##
##     Acf
```

```
df = read_csv("E:\\Stevens institute\\SecondSem\\MA641 Time Series\\Project\\archive\\coin_Cardano.csv"
df = select(df, Date, Close)

df$Date <- as.Date(df$Date, "%Y-%m-%d")
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%Y-%m-%d'
```

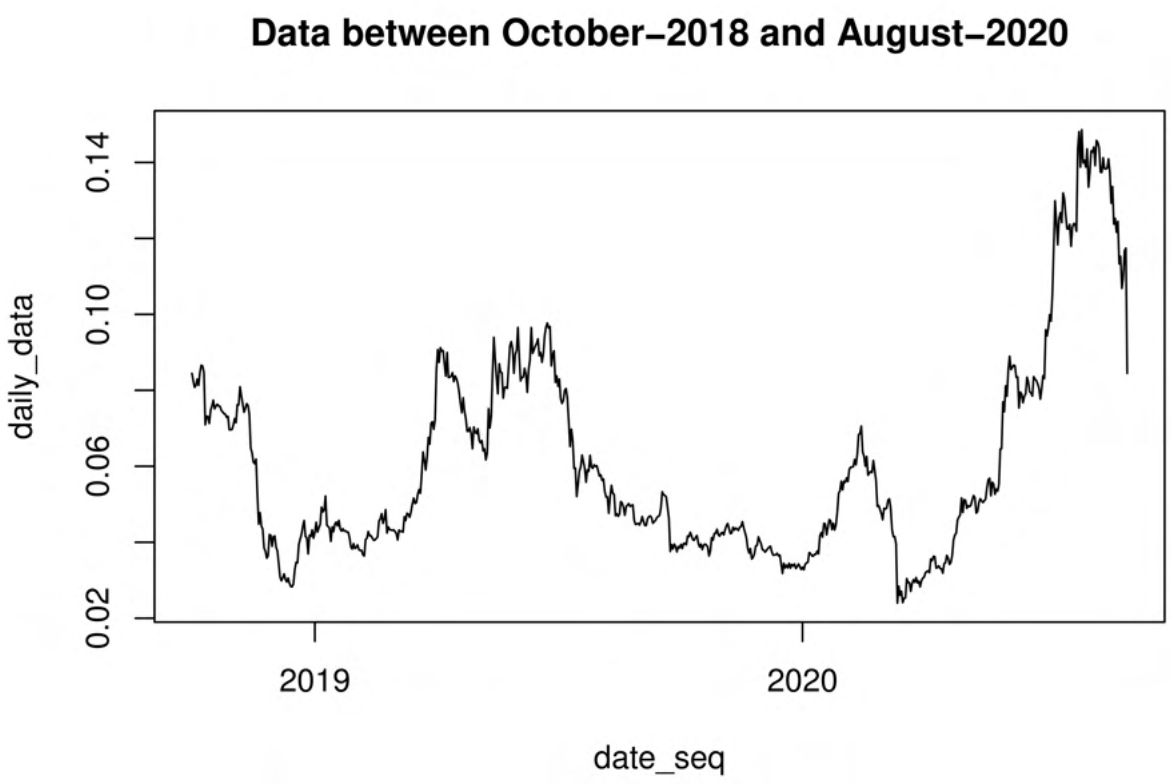
```
# Create a time series with daily data from 2017-10-02 to 2021-07-06
start_date <- as.Date("2017-10-02")
end_date <- as.Date(max(df$Date))
daily_data <- ts(df$Close, start = c(2017, 275), frequency = 365)
plot(daily_data, main = 'original_dataset')
```



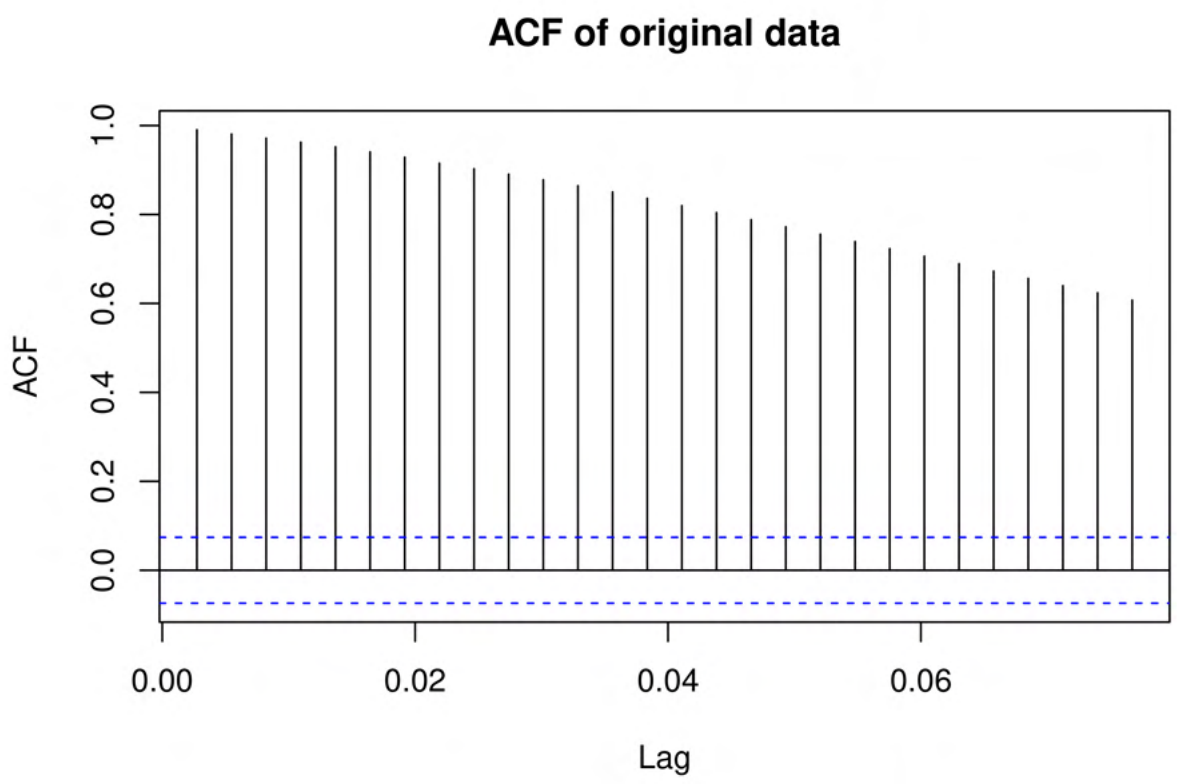

```
##############################################################################################
##############################################################################################

# Select data from 2018 to 2020
start_date <- as.Date("2018-10-01")
end_date <- as.Date("2020-8-31")

########################

#for prediction purpose: ## just next 60 values
df2 =  df[df$Date >= end_date & df$Date <= as.Date("2020-10-31"),]
############################

## for model fitting purpose
df <- df[df$Date >= start_date & df$Date < end_date,]

#####################################################################

# Create a time series with daily data from 2018 to 2020
as.integer(format(as.Date("2018-10-01"), "%j"))
```

```
## [1] 274
```

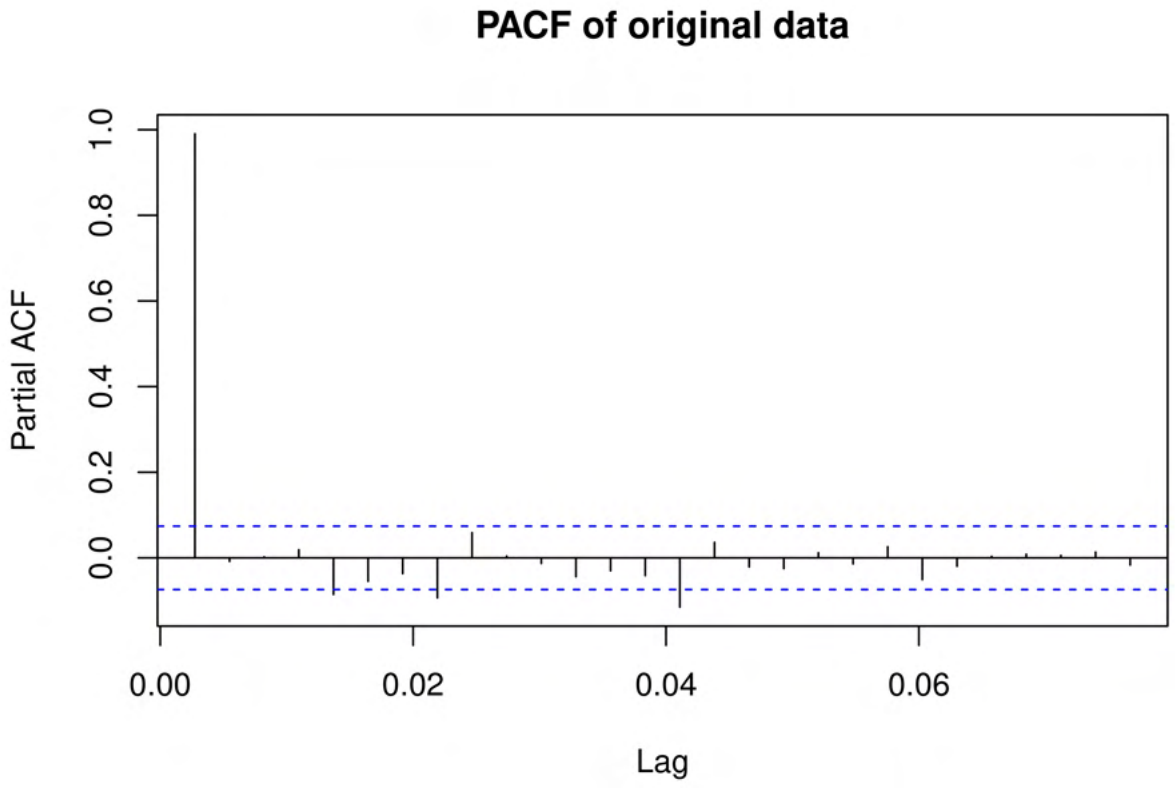
```
daily_data <- ts(df$Close, start = c(2018, 274), end = c(2020, 244), frequency = 365)
date_seq <- seq(from = start_date, to = end_date, by = "day")
plot(date_seq,daily_data, type='l',main='Data between October-2018 and August-2020')
```




```
#####################################################################


df3 <- data.frame(Date = seq(as.Date("2018-10-01"), as.Date("2020-08-31"), by = "day"),
                  Close = daily_data)

#############################################################################

data1 = df3$Close
acf(data1, main = 'ACF of original data')
```



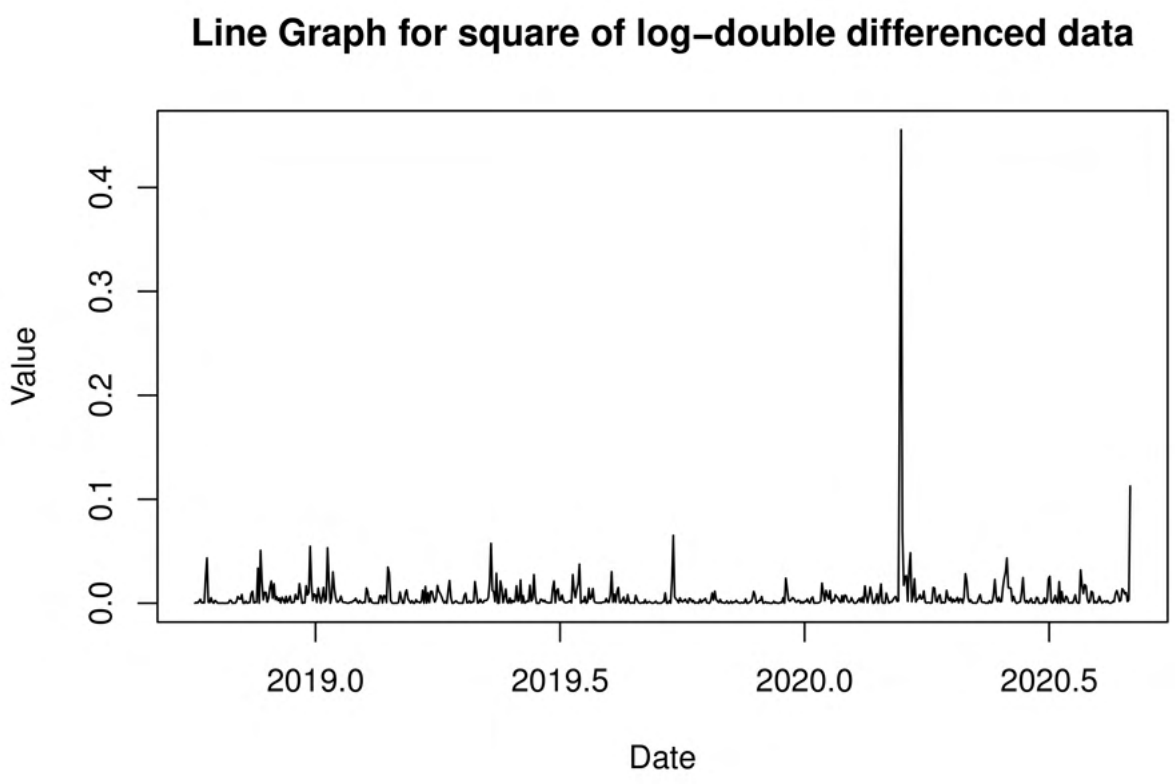

```
## exponentially dying

pacf(data1,main = 'PACF of original data' )
```




```
## Seems like AR(1) model

## Performing dickey Fuller Test

adf_test = adf.test(data1)
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data1
## Dickey-Fuller = -2.2735, Lag order = 8, p-value = 0.4626
## alternative hypothesis: stationary
```

```
## Since p value > 0.05: fail to reject H0: it means non-stationary

######################################################################

## Log differencing

close_diff = (diff(diff(log(data1))))^2

plot(close_diff, type = "l", xlab = "Date", ylab = "Value", main = "Line Graph for square of log-double
```



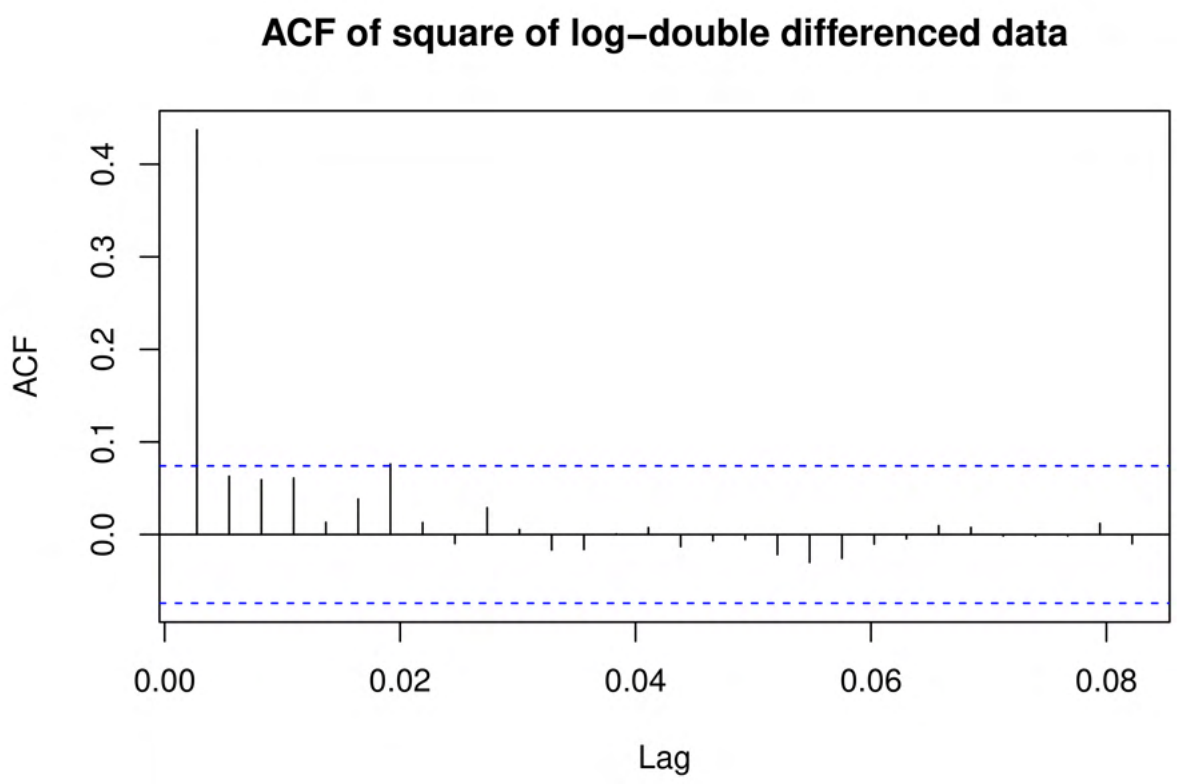

```
adf_test = adf.test(close_diff)
```

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

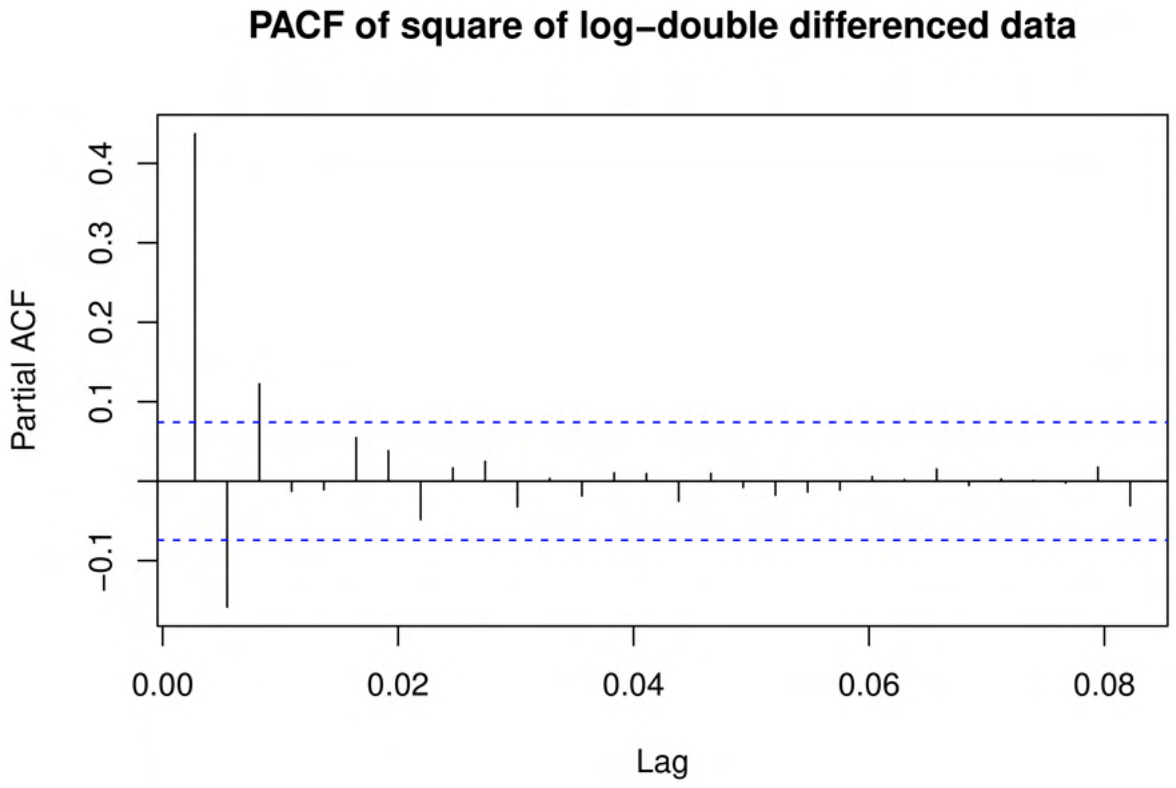
```
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  close_diff
## Dickey-Fuller = -7.5784, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Means the data has become stationary


## acf and pacf plots

acf(close_diff, lag.max = 30, main = 'ACF of square of log-double differenced data')
```




```
pacf(close_diff, lag.max = 30, main = 'PACF of square of log-double differenced data')
```



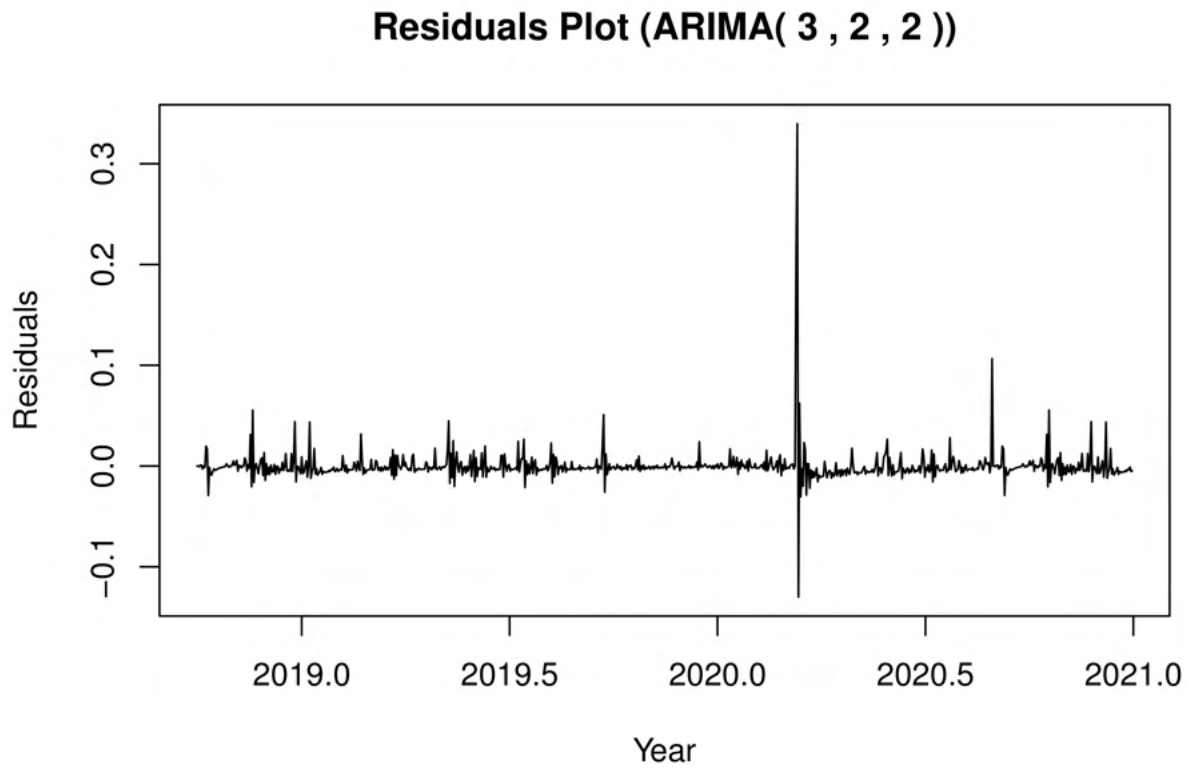

```
eacf(close_diff)
```

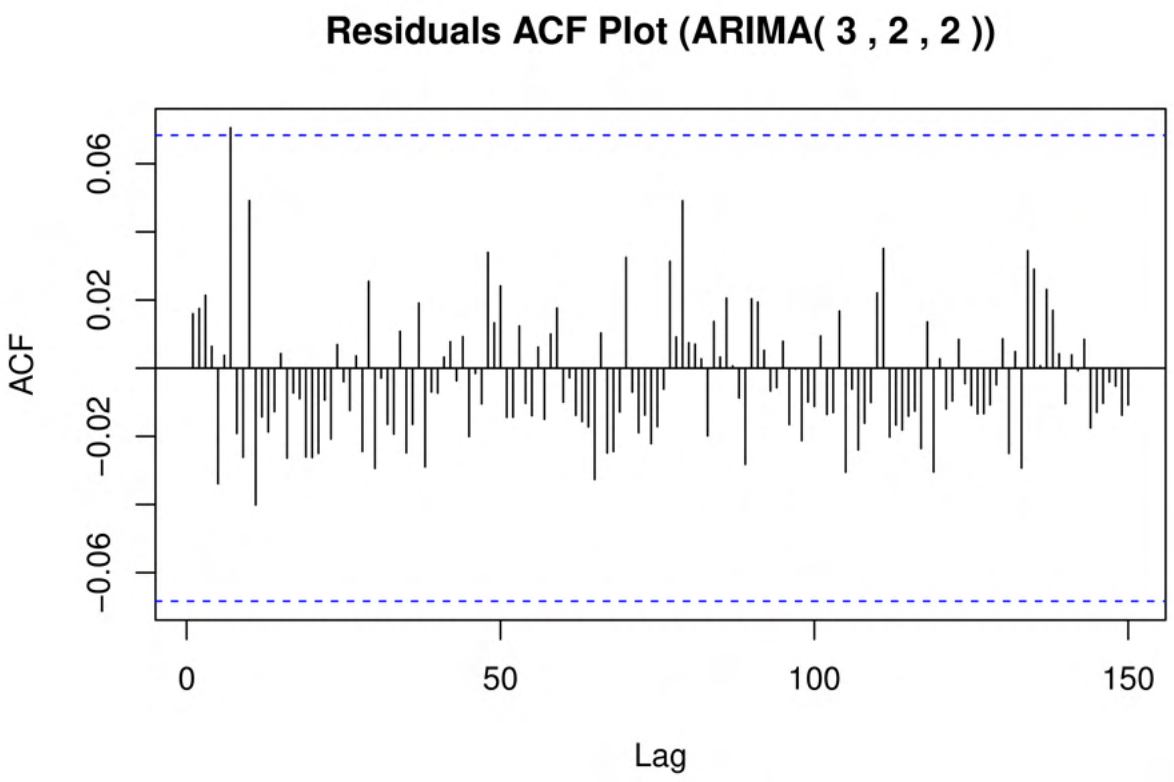
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o x o o o o  o  o  o
## 1 x x o o o o x o o o o  o  o  o
## 2 x x o o o o x o o o o  o  o  o
## 3 x x x x o o o o o o o  o  o  o
## 4 x o x x x o o o o o o  o  o  o
## 5 x x x x x x o o o o o  o  o  o
## 6 x x x x x o x o o o o  o  o  o
## 7 x x x o o o x o o o o  o  o  o
```

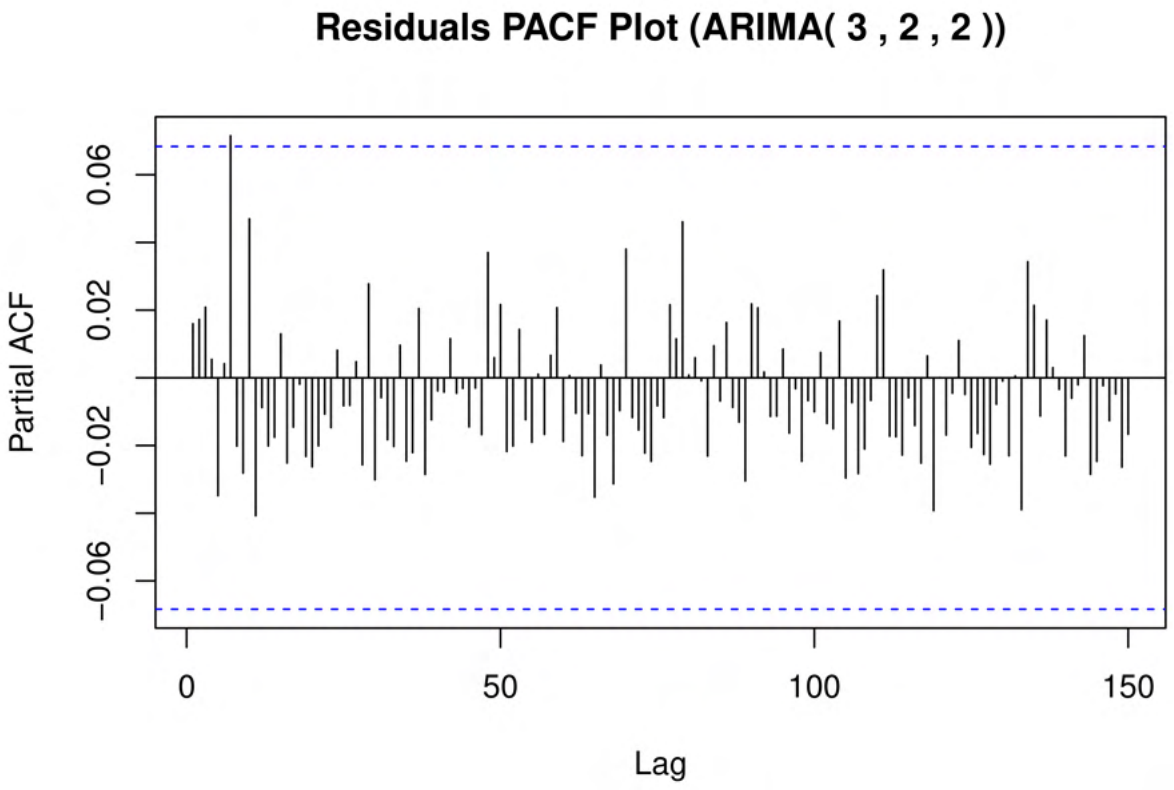
```
##########################################################################################################

## suppose Models:
## ARIMA(3,2,2) ## actually from the model
## ARIMA(0,2,2) ## from the eacfs
## ARIMA(1,2,2)
## ARIMA(3,2,3)
## ARIMA(2,2,3)

##########################################################################################################
##########################################################################################################


## Defining Function:
```

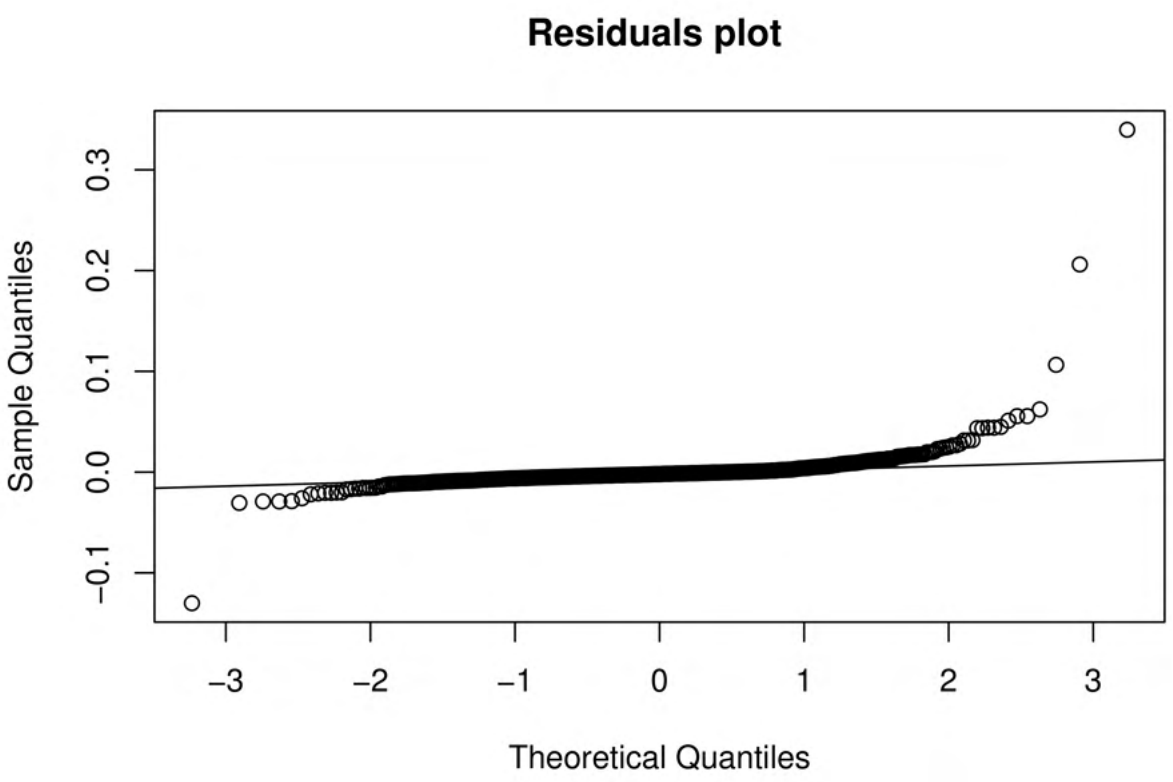
```r
plot_arima_resid <- function(model) {
  # Extract the residuals from the model
  resid <- resid(model)

  # Create a time series of the residuals
  resid_ts <- ts(resid, start = c(2018, 274), end = c(2020, 365), frequency = 365)

  # Get the values of p, d, and q from the ARIMA model
  p <- model$arma[1]
  q <- model$arma[2]

  # Plot the residuals with custom x-axis labels and title with ARIMA values
  plot(resid_ts, xlab = "Year", ylab = "Residuals", main = paste("Residuals Plot (ARIMA(", p, ",", 2, "

  acf(resid_ts[1:length(resid_ts)],lag.max = 150,main = paste("Residuals ACF Plot (ARIMA(", p, ",", 2,
  pacf(resid_ts[1:length(resid_ts)],lag.max = 150, main = paste("Residuals PACF Plot (ARIMA(", p, ",",

}


#################################################################
# create an empty data frame with column names
# create an empty data frame with column names
my_df <- data.frame(model = character(),
                    AIC = numeric(),
                    BIC = numeric(),
                    Shapiro = round(numeric(),3),
                    Ljung = round(numeric(),3))

new_row <- data.frame(model = "", AIC = 0, BIC = 0, Shapiro = 0, Ljung = 0)


plot_arima_residuals <- function(arima_model, my_df) {
  # extract residuals and create time series
  resid <- residuals(arima_model)
  resid_ts <- ts(resid, start = c(2018, 274), end = c(2020, 365), frequency = 365)
  plot_arima_resid(arima_model)

  # normal Q-Q plot of residuals
  qqnorm(resid_ts, main = "Residuals plot ")
  qqline(resid_ts)

  # Shapiro-Wilk test of normality
  shap = shapiro.test(resid_ts)

  # Ljung-Box test of autocorrelation
  ljung = LB.test(arima_model)

  # AIC and BIC
  cat("AIC:", AIC(arima_model), "\n")
  cat("BIC:", BIC(arima_model), "\n")

  # create a new row for the data frame and append it
```

```
  new_row$model <- as.character(arima_model)
  new_row$AIC <- AIC(arima_model)
  new_row$BIC <- BIC(arima_model)
  new_row$Shapiro <- shap$p.value
  new_row$Ljung <- round(ljung$p.value,3)
  my_df <- rbind(my_df, new_row)

  # return the updated data frame
  return(my_df)
}

# call the function and update the data frame
arima322 <- arima(x = close_diff, order = c(3, 2, 2))
arima022 <- arima(x = close_diff, order = c(0, 2, 2))
arima122 <- arima(x = close_diff, order = c(1, 2, 2))
arima323 <- arima(x = close_diff, order = c(3, 2, 3))
arima223 <- arima(x = close_diff, order = c(2, 2, 3))



my_df <- plot_arima_residuals(arima322, my_df)
```



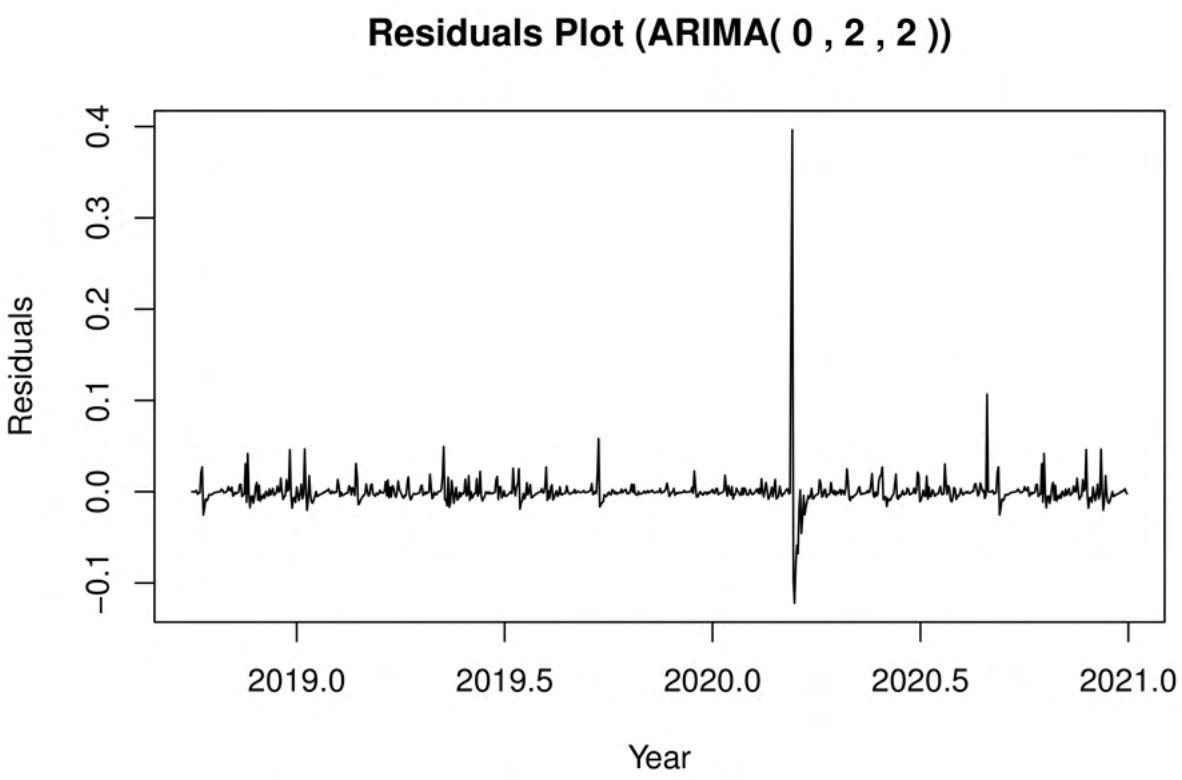

Residuals ACF Plot (ARIMA(3 , 2 , 2))
ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag

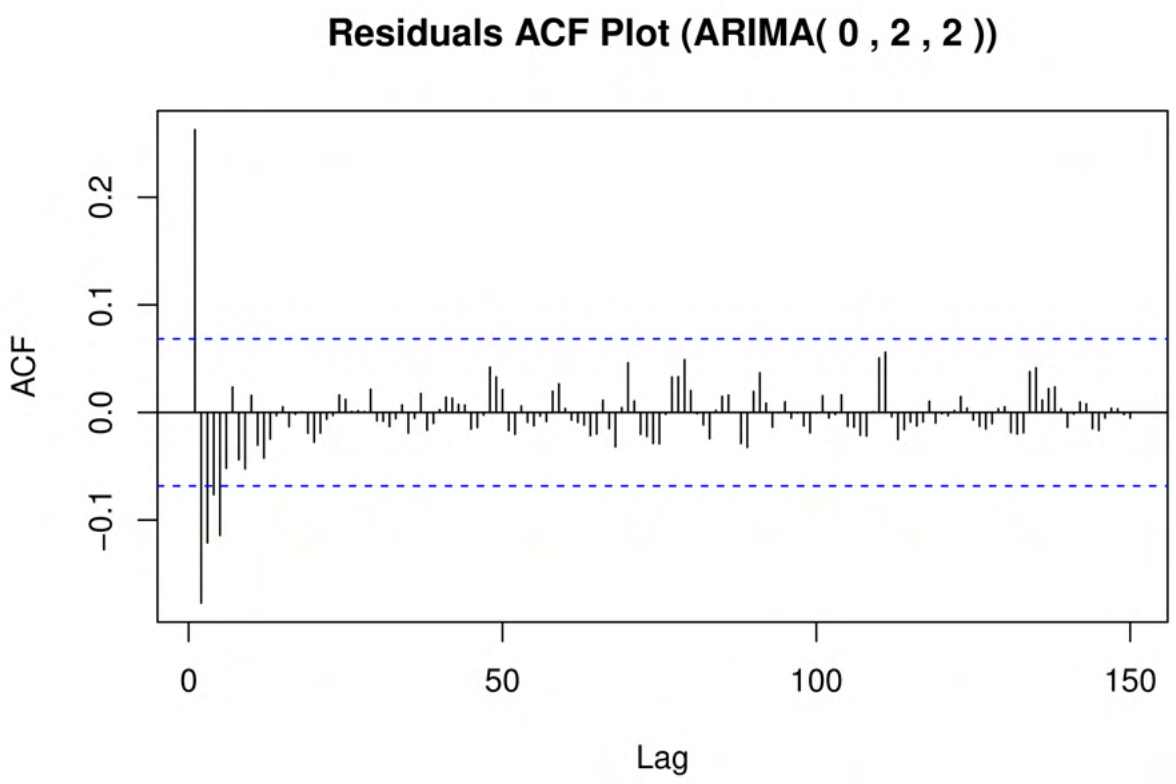

Residuals PACF Plot (ARIMA(3 , 2 , 2))
Partial ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag

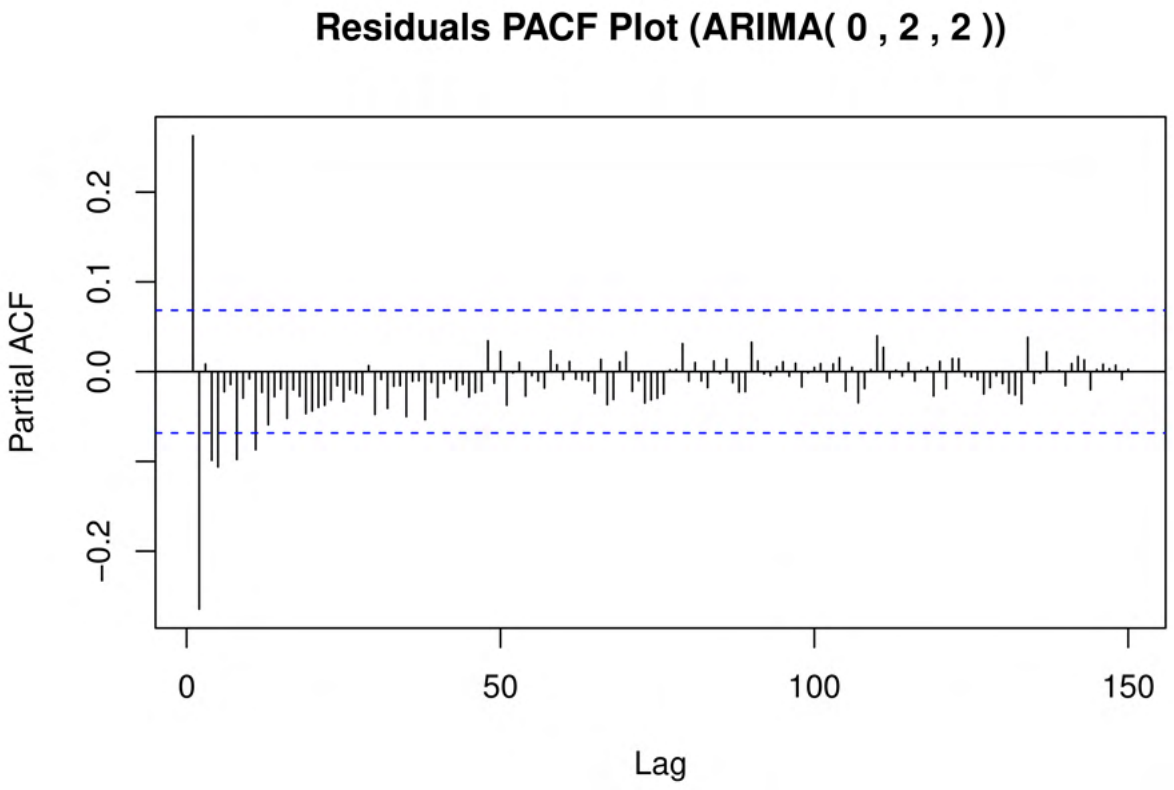


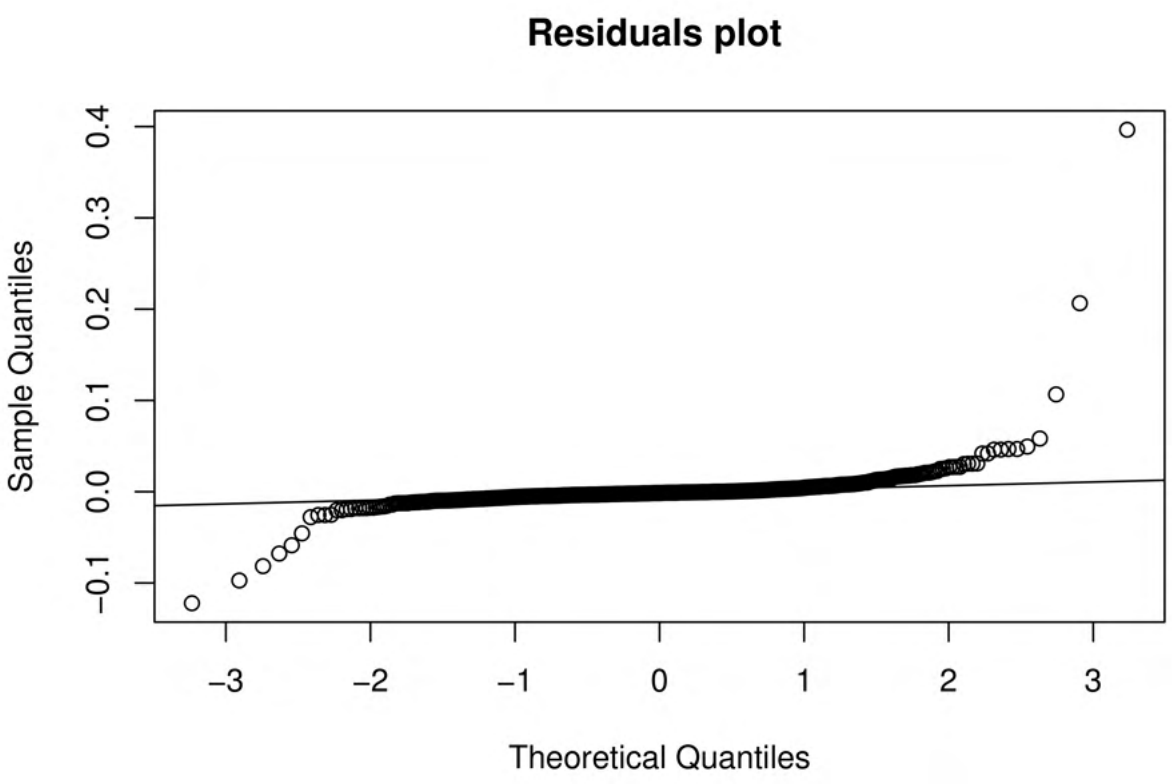

```
## AIC: -3559.729
## BIC: -3532.449
```

```
my_df <- plot_arima_residuals(arima022, my_df)
```


Residuals Plot (ARIMA(0 , 2 , 2))
Residuals
−0.1
0.0
0.1
0.2
0.3
0.4
2019.0
2019.5
2020.0
2020.5
2021.0
Year

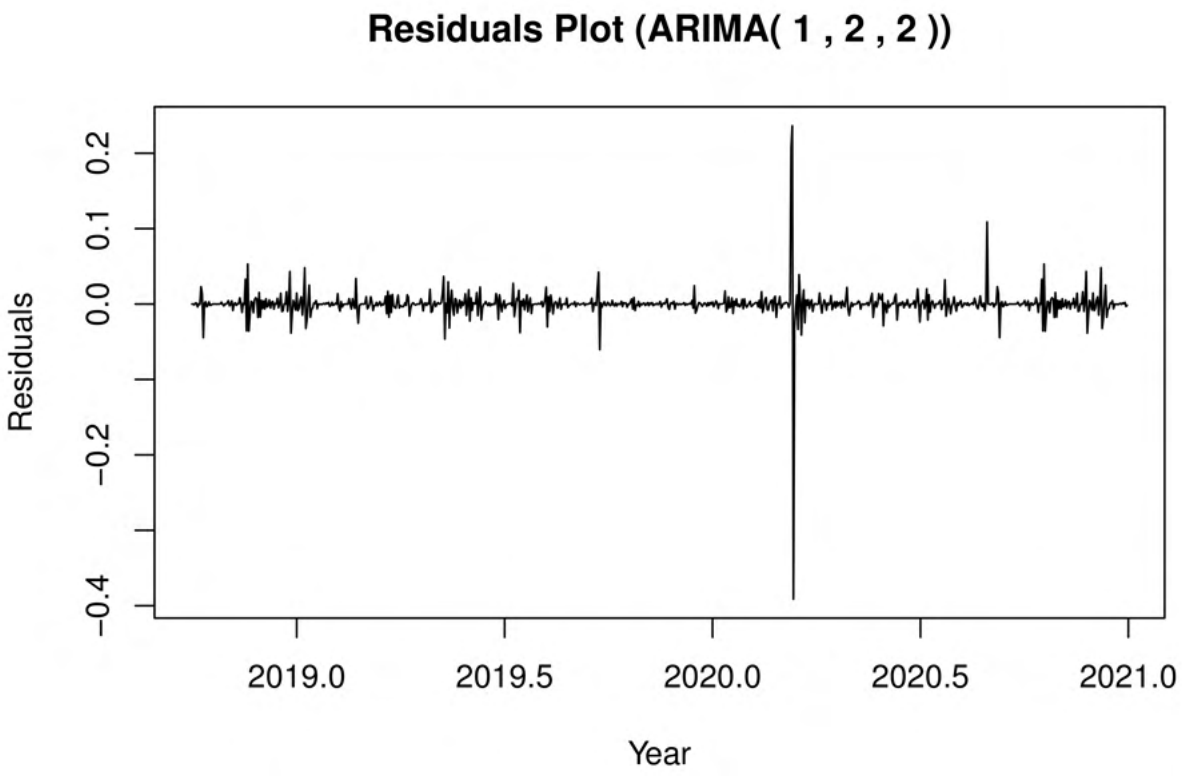

Residuals ACF Plot (ARIMA(0 , 2 , 2))
ACF
0.2
0.1
0.0
−0.1
0
50
100
150
Lag

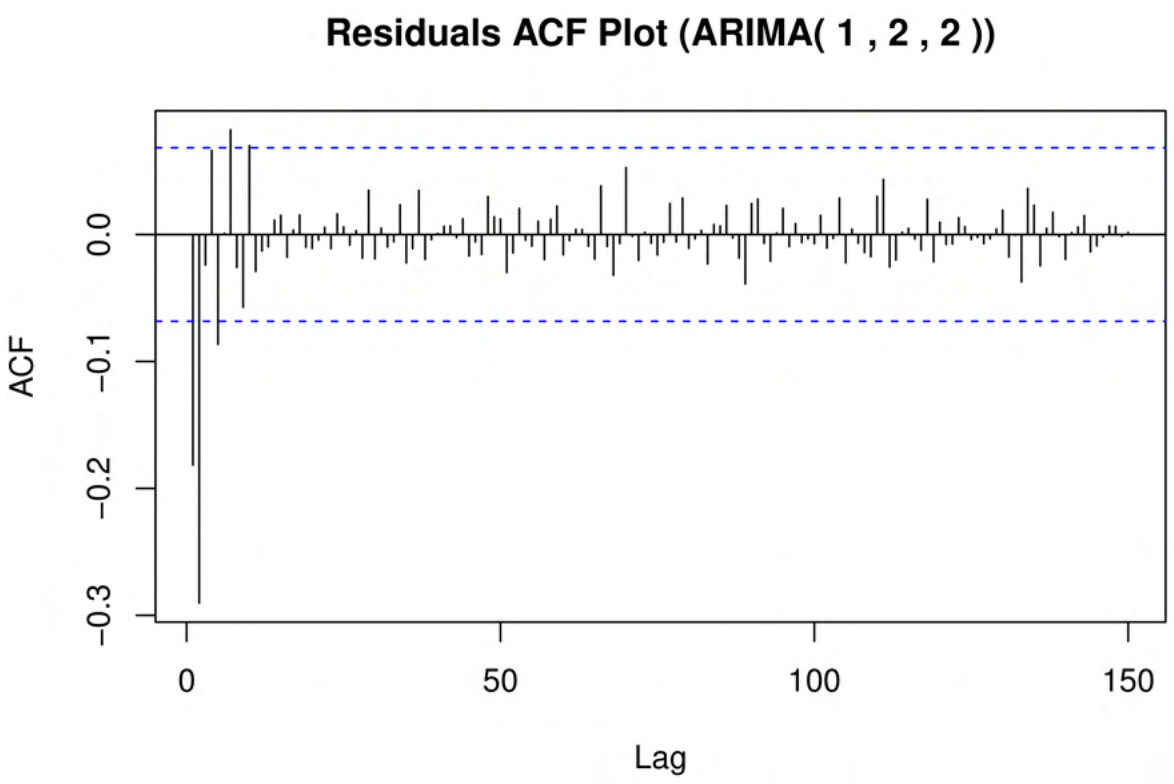

Residuals PACF Plot (ARIMA(0 , 2 , 2))
Partial ACF
0.2
0.1
0.0
−0.2
0
50
100
150
Lag

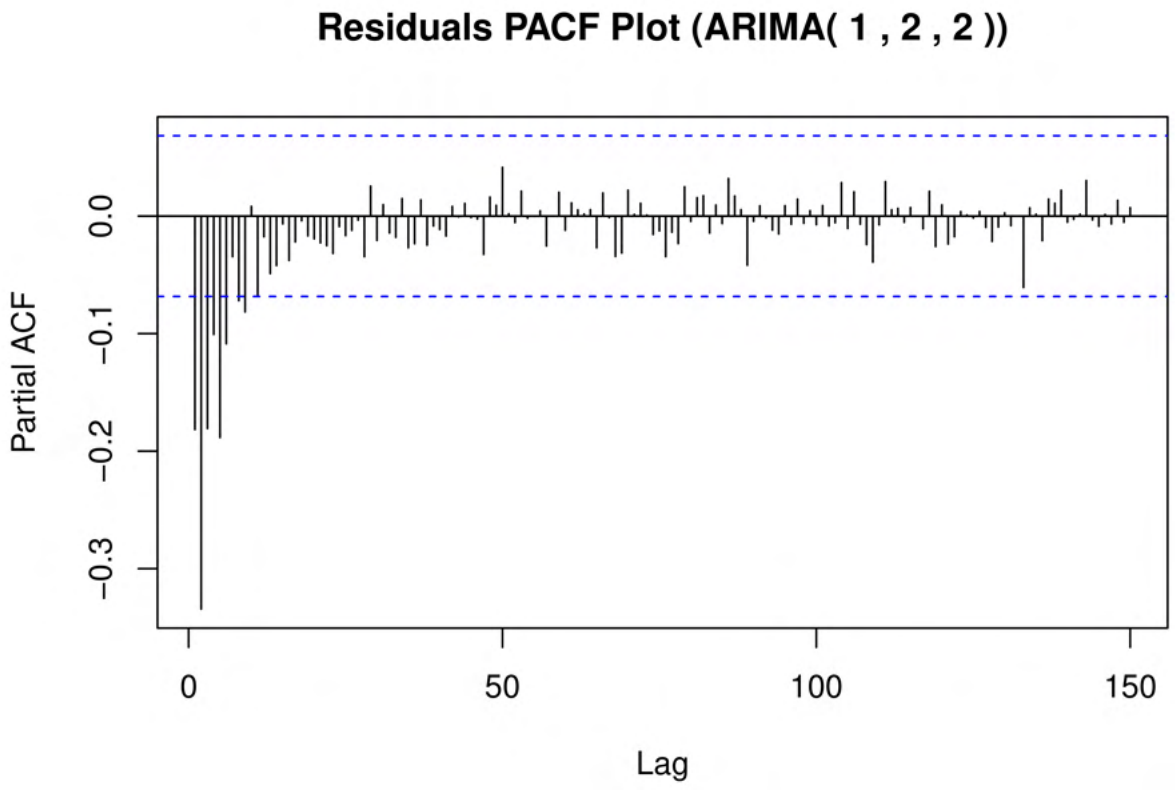


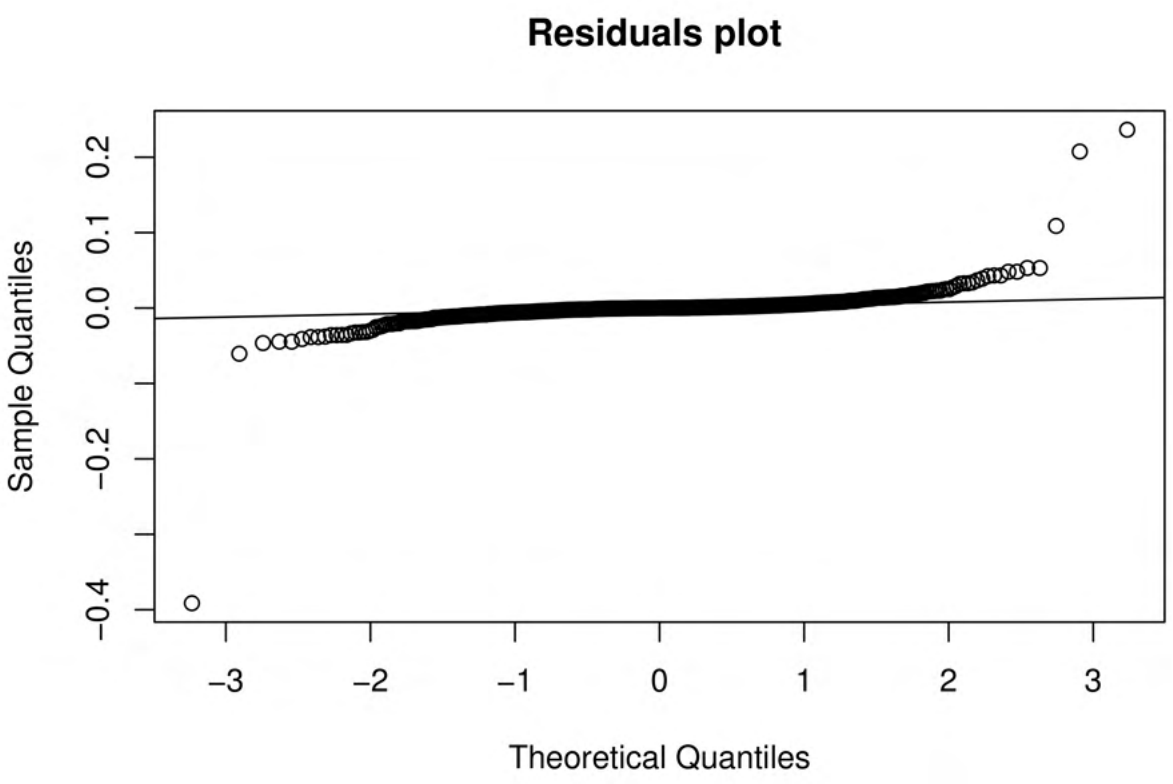

```
## AIC: -3409.742
## BIC: -3396.101
```

```
my_df <- plot_arima_residuals(arima122, my_df)
```


Residuals Plot (ARIMA(1 , 2 , 2))
Residuals
0.2
0.1
0.0
−0.2
−0.4
2019.0
2019.5
2020.0
2020.5
2021.0
Year

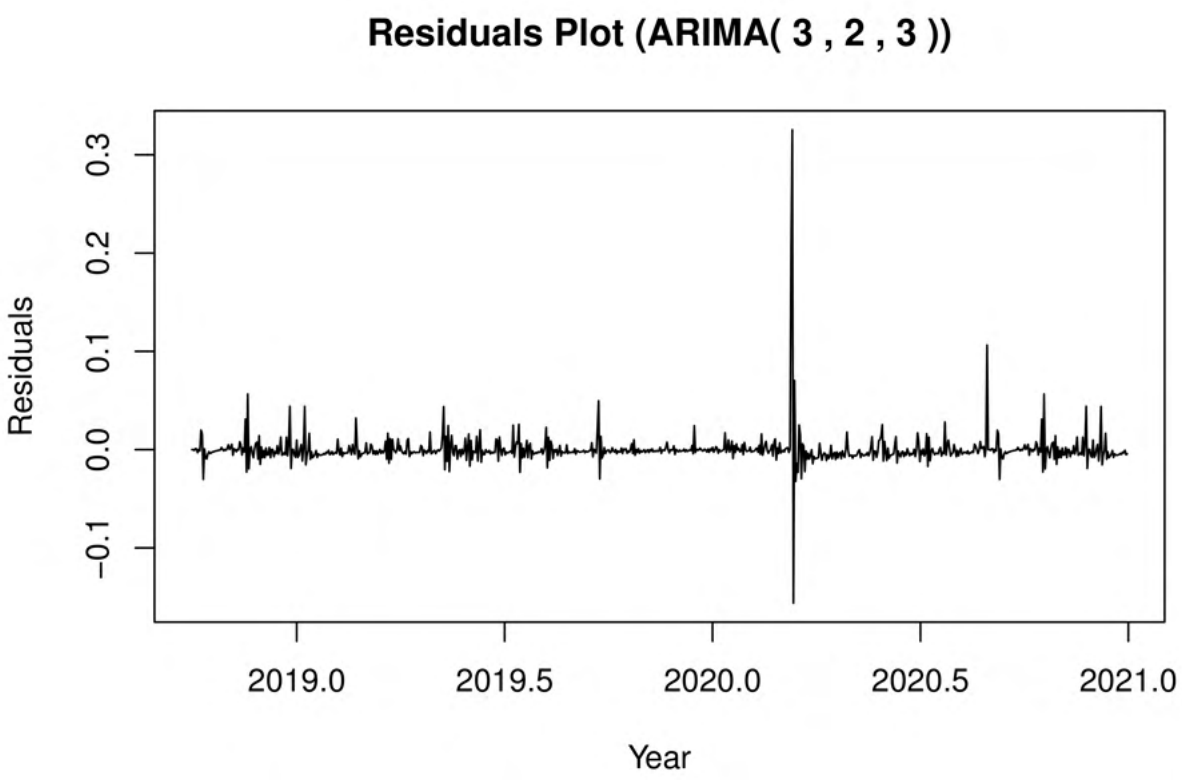

Residuals ACF Plot (ARIMA(1 , 2 , 2))
ACF
0.0
−0.1
−0.2
−0.3
0
50
100
150
Lag

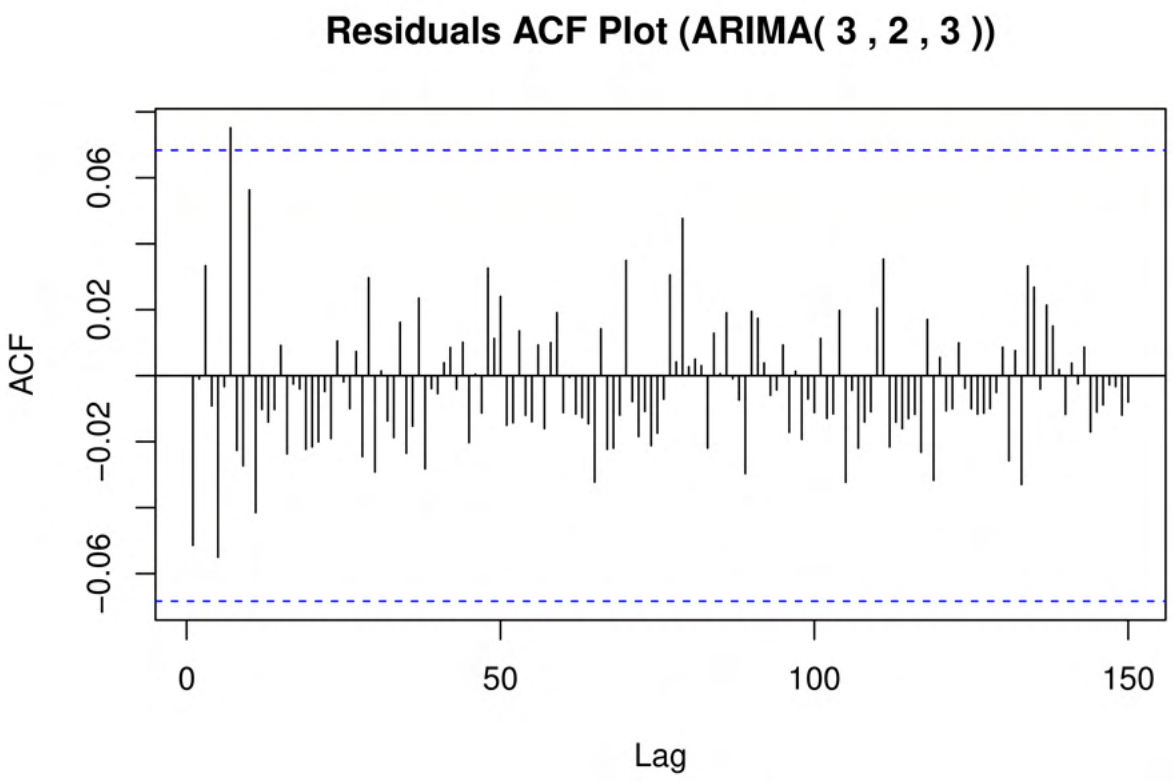

Residuals PACF Plot (ARIMA(1 , 2 , 2))
Partial ACF
0.0
−0.1
−0.2
−0.3
0
50
100
150
Lag

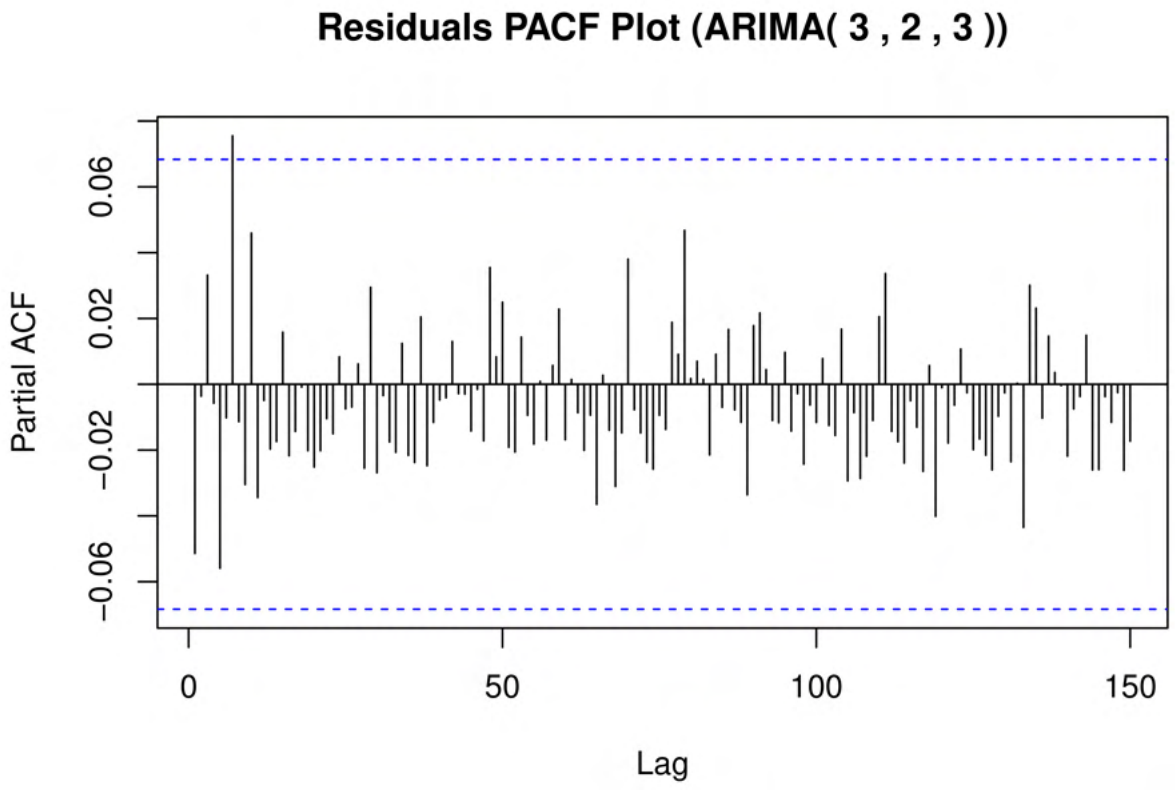


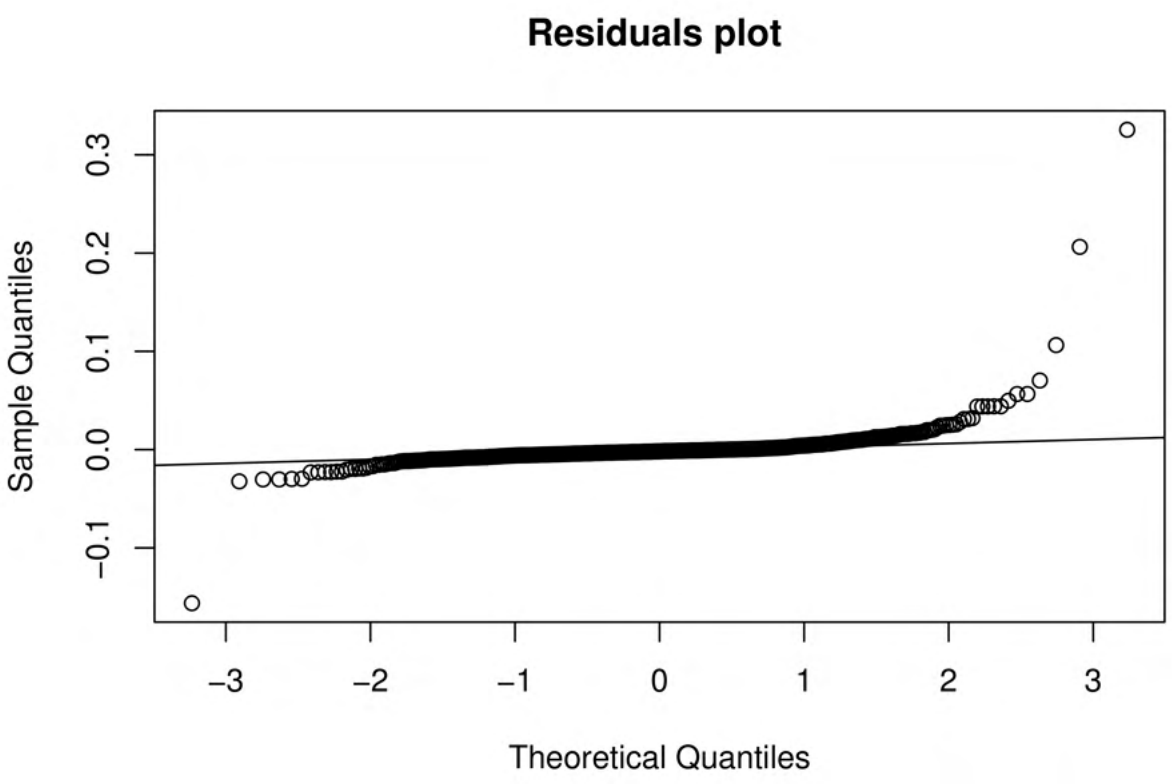

```
## AIC: -3336.977
## BIC: -3318.79
```

```
my_df <- plot_arima_residuals(arima323, my_df)
```


Residuals Plot (ARIMA(3 , 2 , 3))
Residuals
0.3
0.2
0.1
0.0
−0.1
2019.0
2019.5
2020.0
2020.5
2021.0
Year

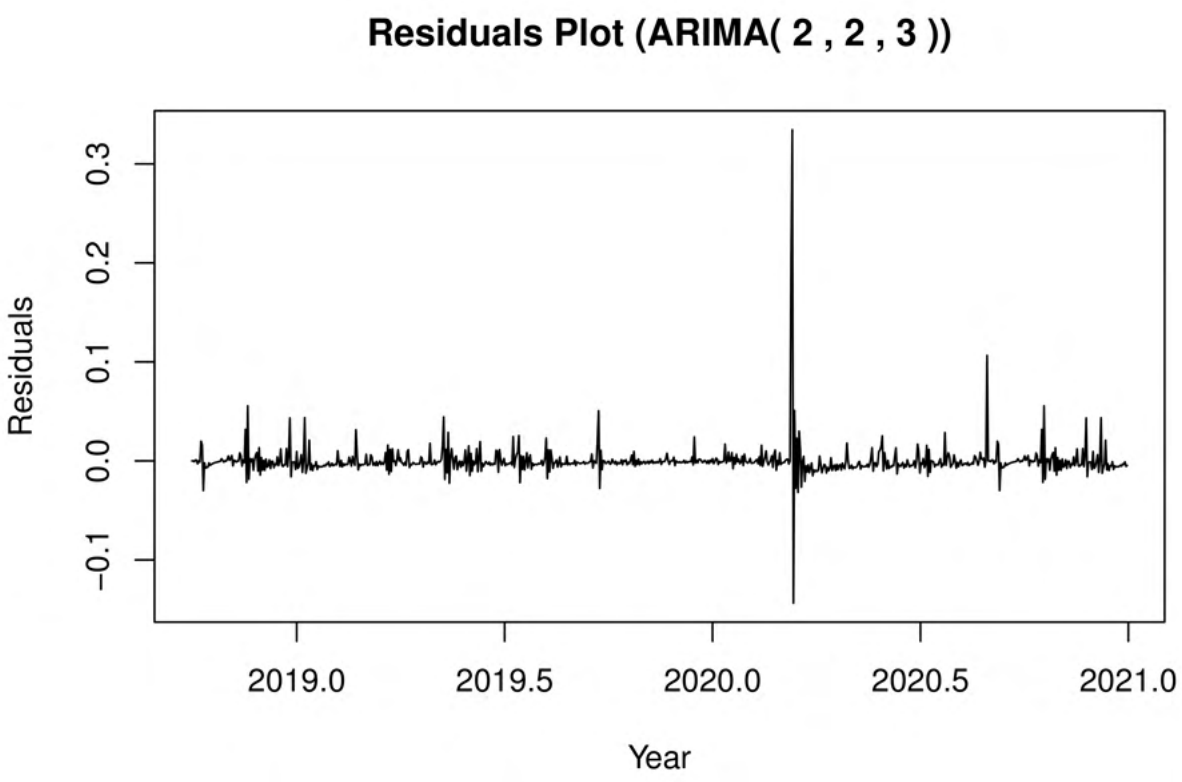

Residuals ACF Plot (ARIMA(3 , 2 , 3))
ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag

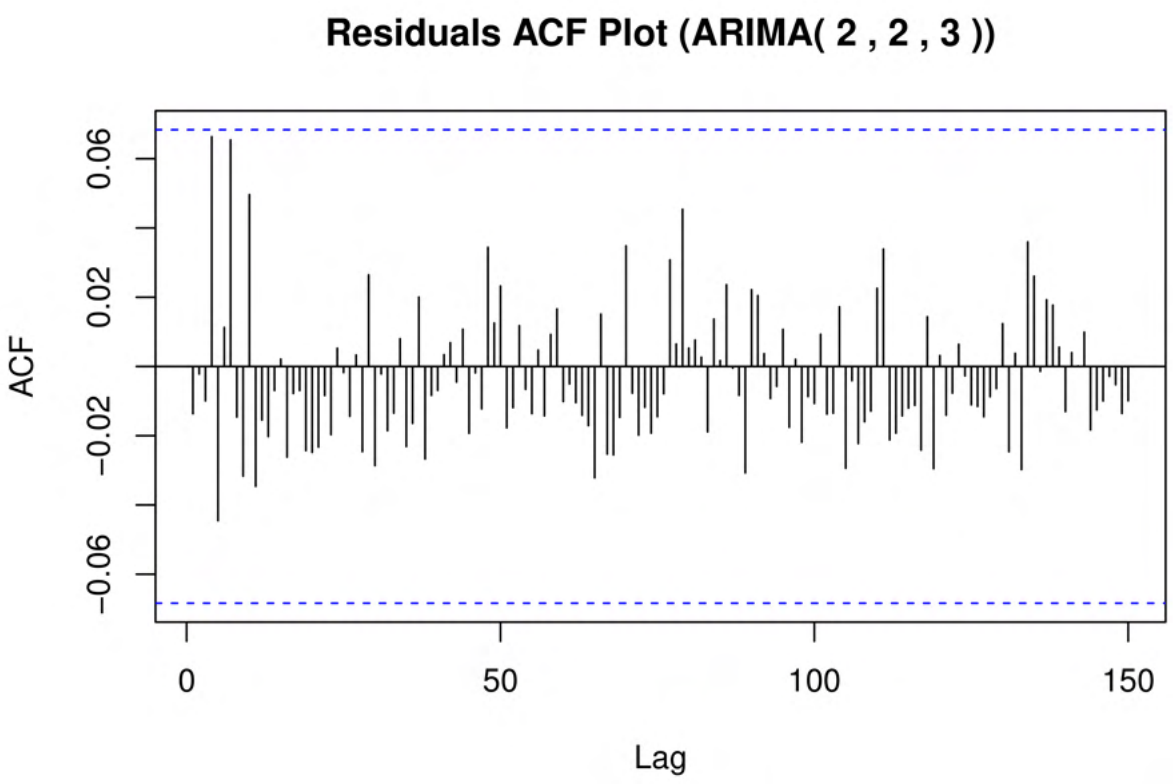

Residuals PACF Plot (ARIMA(3 , 2 , 3))
Partial ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag

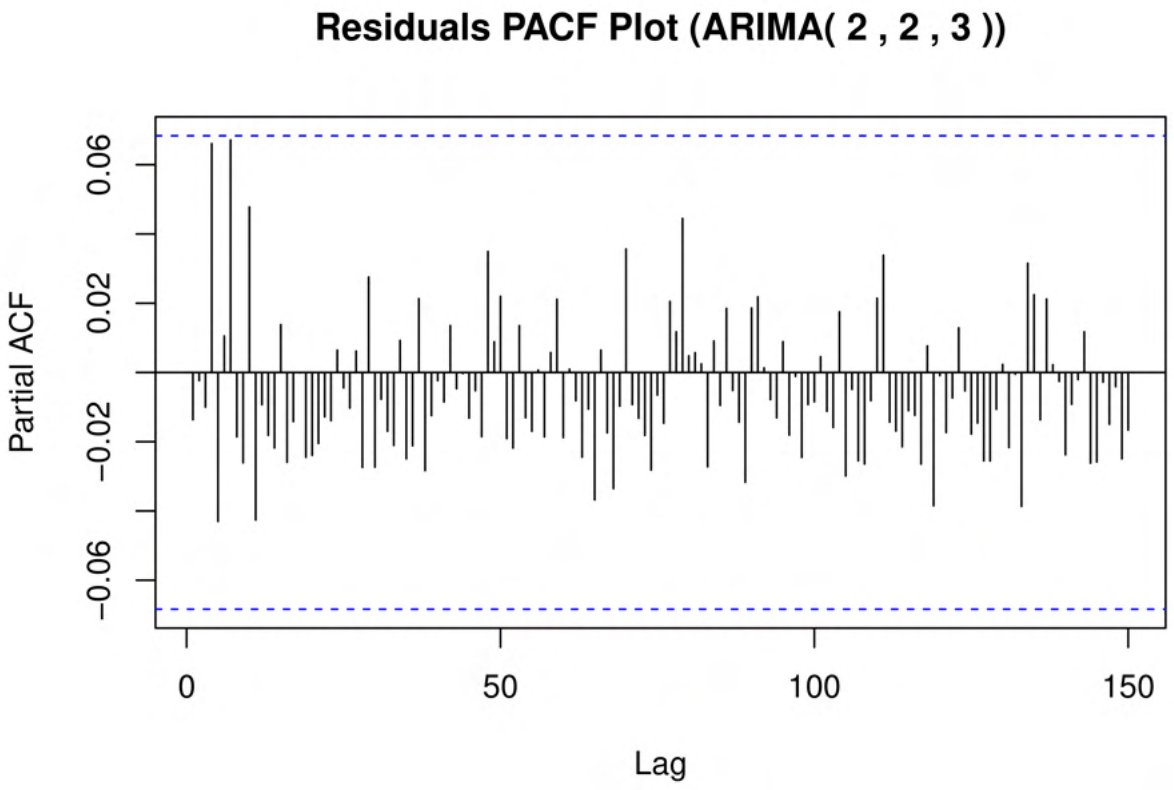


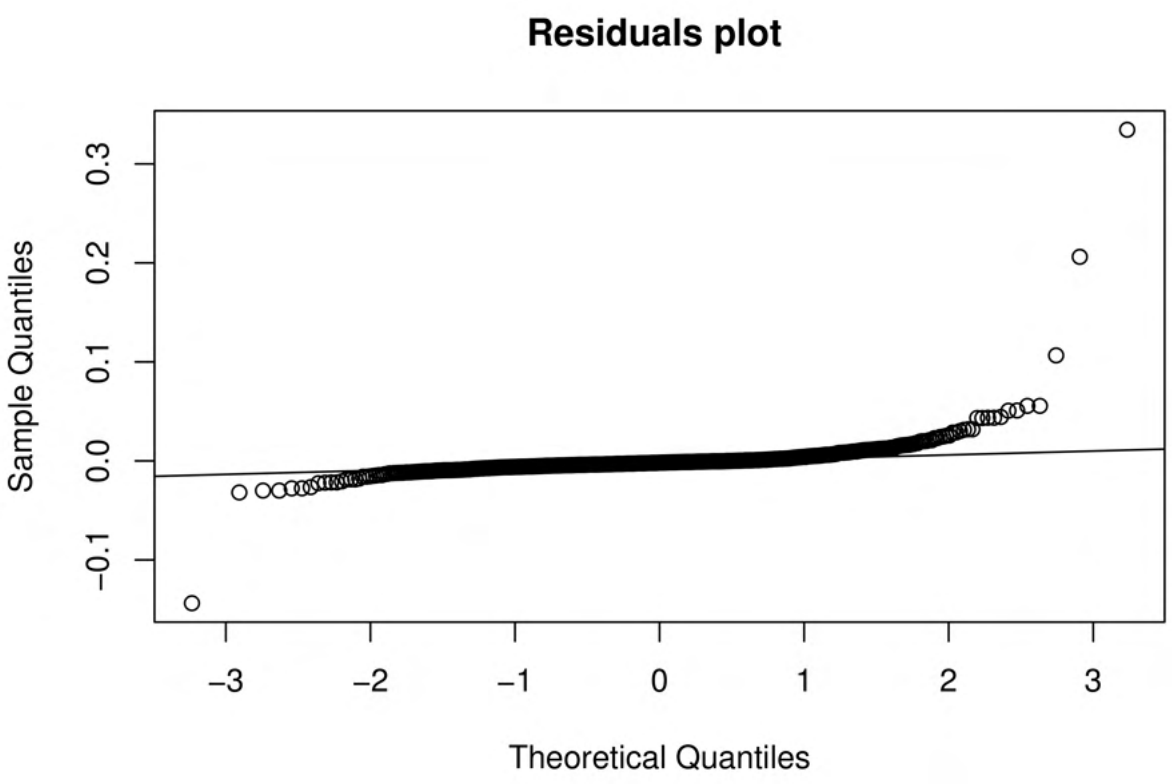

```
## AIC: -3556.466
## BIC: -3524.639
```

```
my_df <- plot_arima_residuals(arima223, my_df)
```


Residuals Plot (ARIMA(2 , 2 , 3))
Residuals
−0.1
0.0
0.1
0.2
0.3
2019.0
2019.5
2020.0
2020.5
2021.0
Year


Residuals ACF Plot (ARIMA(2 , 2 , 3))
ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag


Residuals PACF Plot (ARIMA(2 , 2 , 3))
Partial ACF
0.06
0.02
−0.02
−0.06
0
50
100
150
Lag




```
## AIC: -3558.017
## BIC: -3530.736
```

```
# view updated data frame
print(my_df)
```

```
##          model       AIC       BIC      Shapiro Ljung
## 1 ARIMA(3,2,2) -3559.729 -3532.449 1.556828e-46 0.341
## 2 ARIMA(0,2,2) -3409.742 -3396.101 8.646146e-47 0.000
## 3 ARIMA(1,2,2) -3336.977 -3318.790 4.176565e-46 0.000
## 4 ARIMA(3,2,3) -3556.466 -3524.639 3.768687e-46 0.099
## 5 ARIMA(2,2,3) -3558.017 -3530.736 2.529240e-46 0.213
```

```
################################################################################
#########################################################################

cat("## Garch Model ")
```

```
## ## Garch Model
```

```
return = diff(log(df3$Close))

ArchTest(return)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  return
## Chi-squared = 17.207, df = 12, p-value = 0.142
```

```
## No ARCH EFFECT is there

 ##########################################################################
##########################################################################


## finding the GARCH order

garch(x=return,grad = 'numerical', trace = FALSE)
```

```
##
## Call:
## garch(x = return, grad = "numerical", trace = FALSE)
##
## Coefficient(s):
##       a0        a1        b1
## 0.002689  0.050000  0.050000
```

```
## a1 and b1: MEANS : alpha = 1 and beta = 1
## GARCH(1,1)

##########################################################################
##########################################################################

## Garch Model - Normal
## Normal: Pearson value < 0.05: not good
## sstd: pearson > 0.05: good

library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 4.2.3
```

```
## Loading required package: parallel
##
## Attaching package: 'rugarch'
##
## The following object is masked from 'package:purrr':
##
##     reduce
##
## The following object is masked from 'package:stats':
##
##     sigma
```

```
## A) using Normal distribution model

return = diff(log(df3$Close))

s_norm <- ugarchspec(mean.model = list(armaOrder = c(3,2)),variance.model = list(garchOrder = c(1,1)),
                     distribution.model = 'sstd')

m_norm <- ugarchfit(data = return, spec = s_norm, solver ='hybrid')
m_norm
```

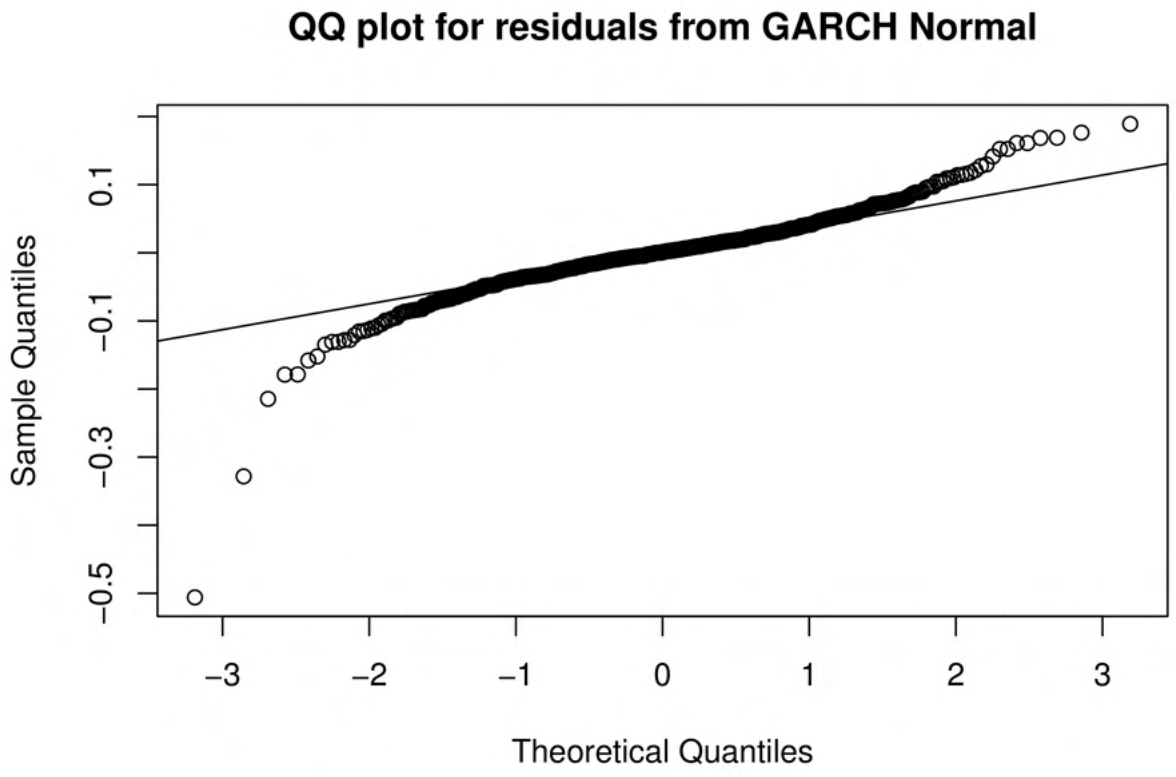
```
##
## *---------------------------------*
```

```
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ---------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000156    0.002024 -0.076878 0.938720
## ar1     0.383922    0.397342  0.966227 0.333930
## ar2     0.438508    0.304523  1.439983 0.149872
## ar3     0.033814    0.063401  0.533335 0.593802
## ma1    -0.506525    0.396307 -1.278112 0.201210
## ma2    -0.342009    0.352932 -0.969051 0.332520
## omega   0.000429    0.000187  2.292001 0.021906
## alpha1  0.130492    0.058347  2.236495 0.025319
## beta1   0.777131    0.069215 11.227800 0.000000
## skew    0.976442    0.049275 19.816091 0.000000
## shape   3.048401    0.418177  7.289738 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000156    0.002195 -0.070908 0.943471
## ar1     0.383922    0.096069  3.996334 0.000064
## ar2     0.438508    0.097503  4.497395 0.000007
## ar3     0.033814    0.029899  1.130940 0.258080
## ma1    -0.506525    0.095816 -5.286426 0.000000
## ma2    -0.342009    0.101531 -3.368520 0.000756
## omega   0.000429    0.000124  3.451614 0.000557
## alpha1  0.130492    0.048416  2.695198 0.007035
## beta1   0.777131    0.043756 17.760681 0.000000
## skew    0.976442    0.051900 18.813896 0.000000
## shape   3.048401    0.403132  7.561794 0.000000
##
## LogLikelihood : 1145.442
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.2413
## Bayes        -3.1697
## Shibata      -3.2417
## Hannan-Quinn -3.2136
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                             statistic p-value
## Lag[1]                          2.714 0.09948
## Lag[2*(p+q)+(p+q)-1][14]        6.913 0.83702
## Lag[4*(p+q)+(p+q)-1][24]       10.629 0.74966
```

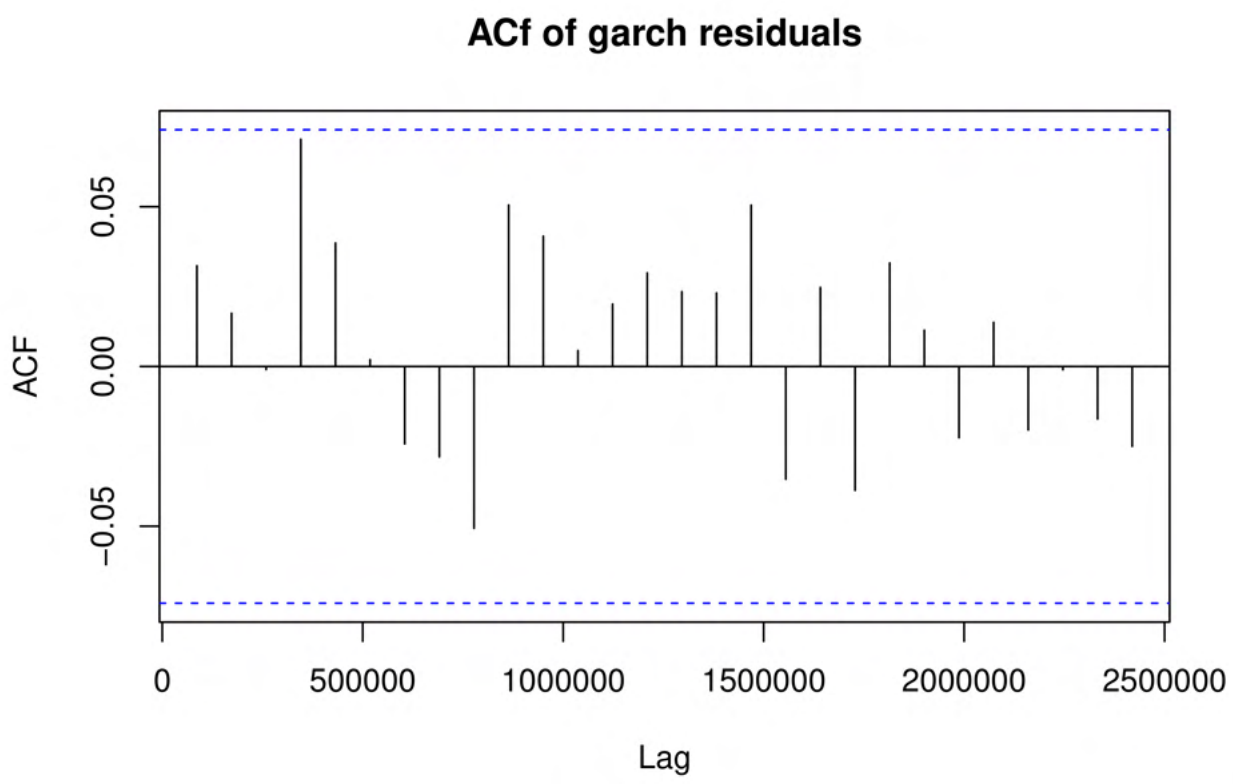
```
## d.o.f=5
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1491  0.6994
## Lag[2*(p+q)+(p+q)-1][5]    1.0300  0.8529
## Lag[4*(p+q)+(p+q)-1][9]    1.7129  0.9364
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1013 0.500 2.000  0.7503
## ARCH Lag[5]    1.4514 1.440 1.667  0.6052
## ARCH Lag[7]    1.6245 2.315 1.543  0.7961
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.6203
## Individual Statistics:
## mu     0.24498
## ar1    0.12582
## ar2    0.07218
## ar3    0.09464
## ma1    0.12610
## ma2    0.07947
## omega  0.09535
## alpha1 0.16483
## beta1  0.10484
## skew   0.22460
## shape  0.10170
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.49 2.75 3.27
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value   prob sig
## Sign Bias          0.008354 0.9933
## Negative Sign Bias 0.090984 0.9275
## Positive Sign Bias 0.665173 0.5062
## Joint Effect       0.748124 0.8618
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     10.57       0.9374
## 2    30     32.00       0.3199
## 3    40     27.77       0.9101
## 4    50     41.71       0.7604
```

```
##
##
## Elapsed time : 0.4915371
```

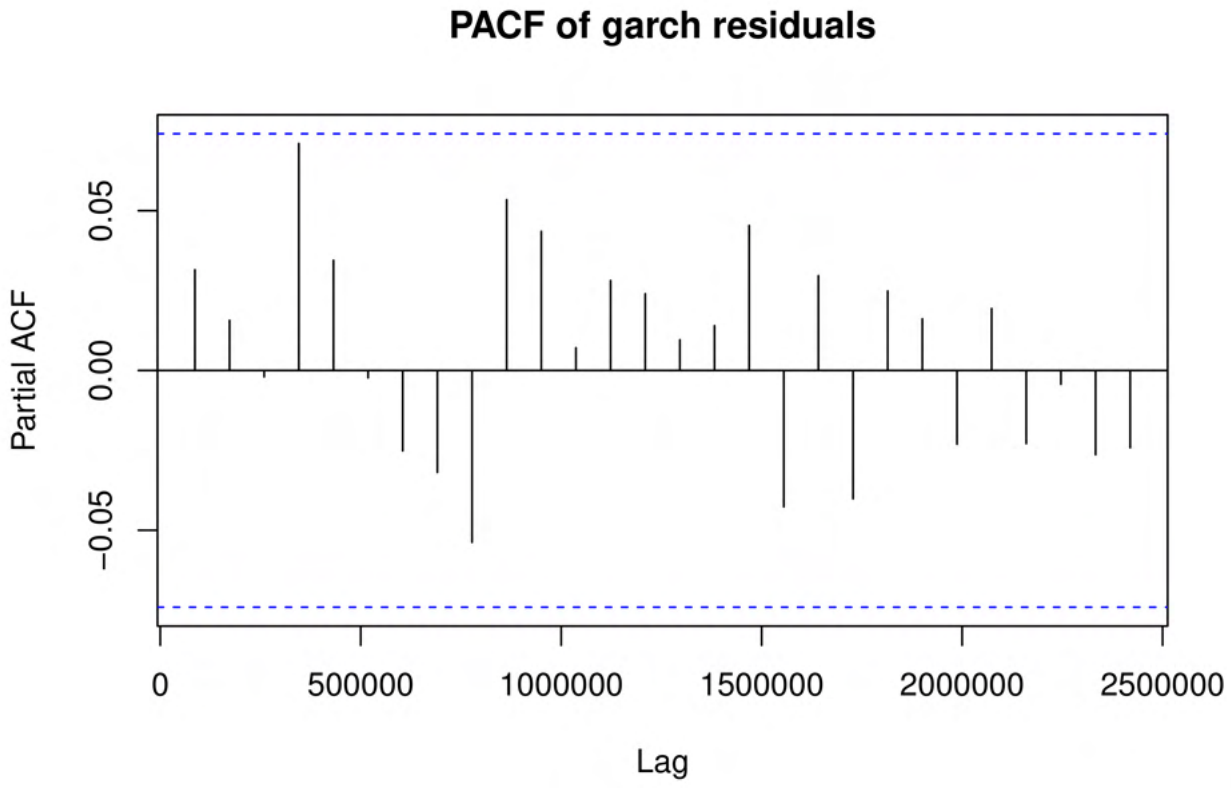
```
resid_norm = residuals(m_norm)
qqnorm(resid_norm,  main = 'QQ plot for residuals from GARCH Normal')
qqline(resid_norm)
```



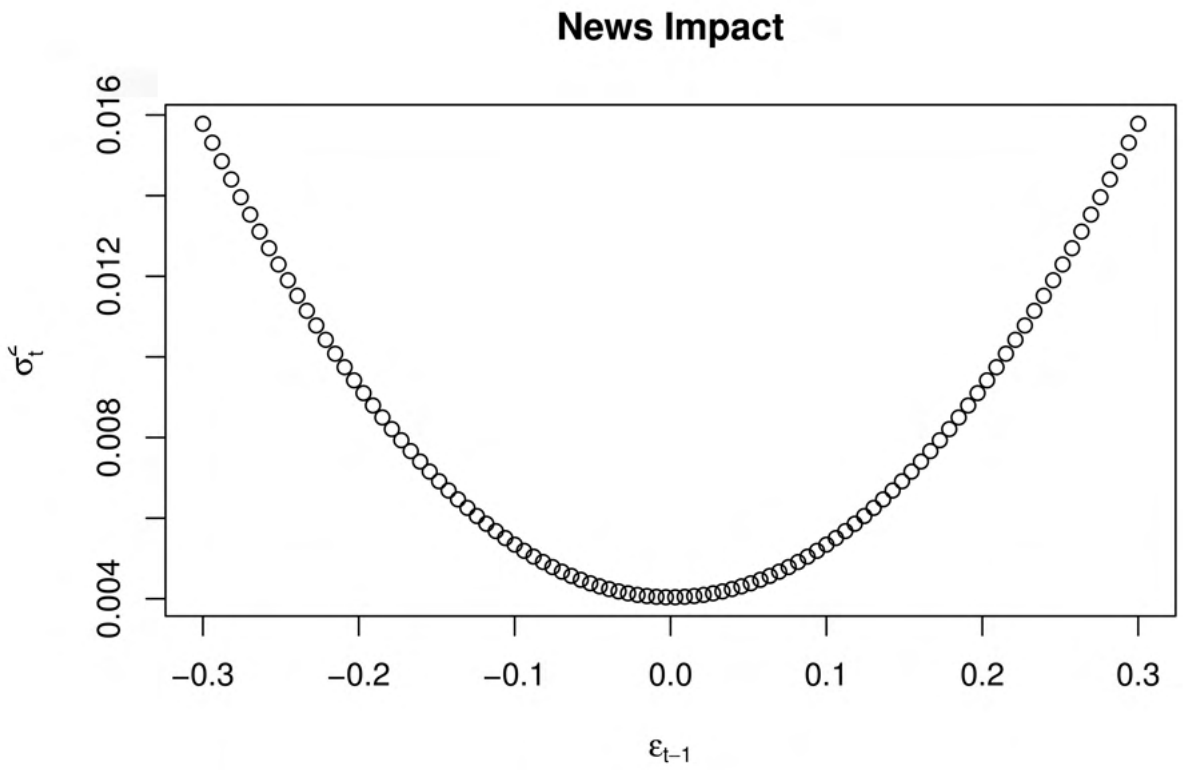

```
acf(resid_norm, main="ACf of garch residuals")
```



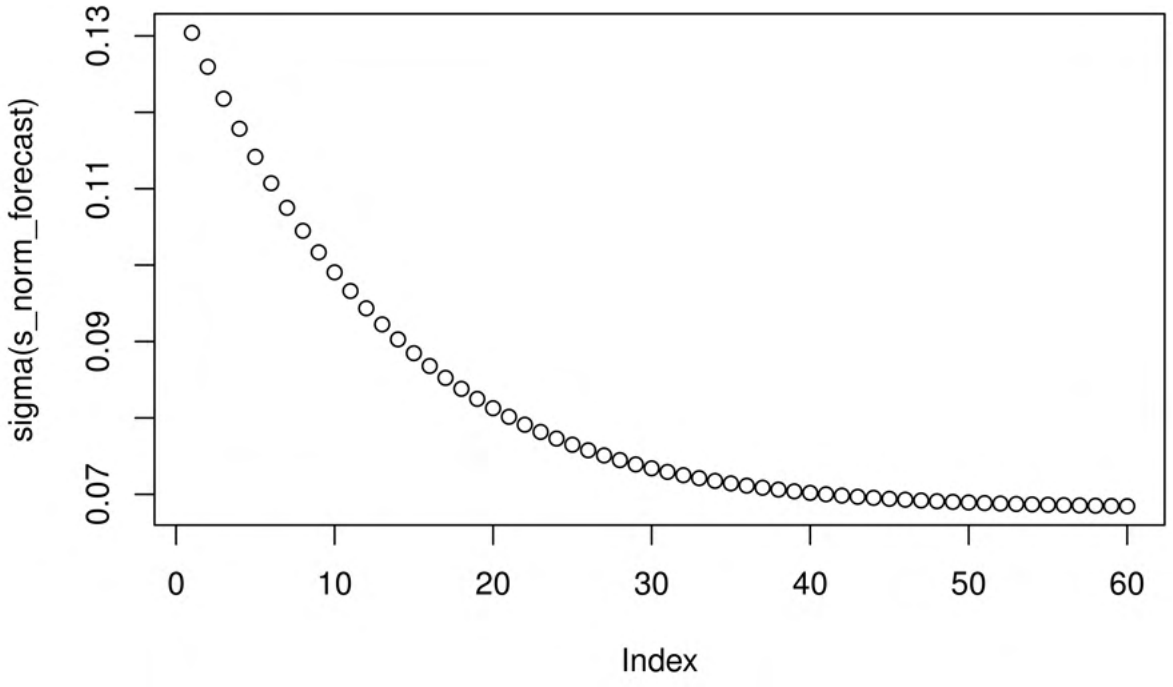

```
pacf(resid_norm, main = "PACF of garch residuals")
```



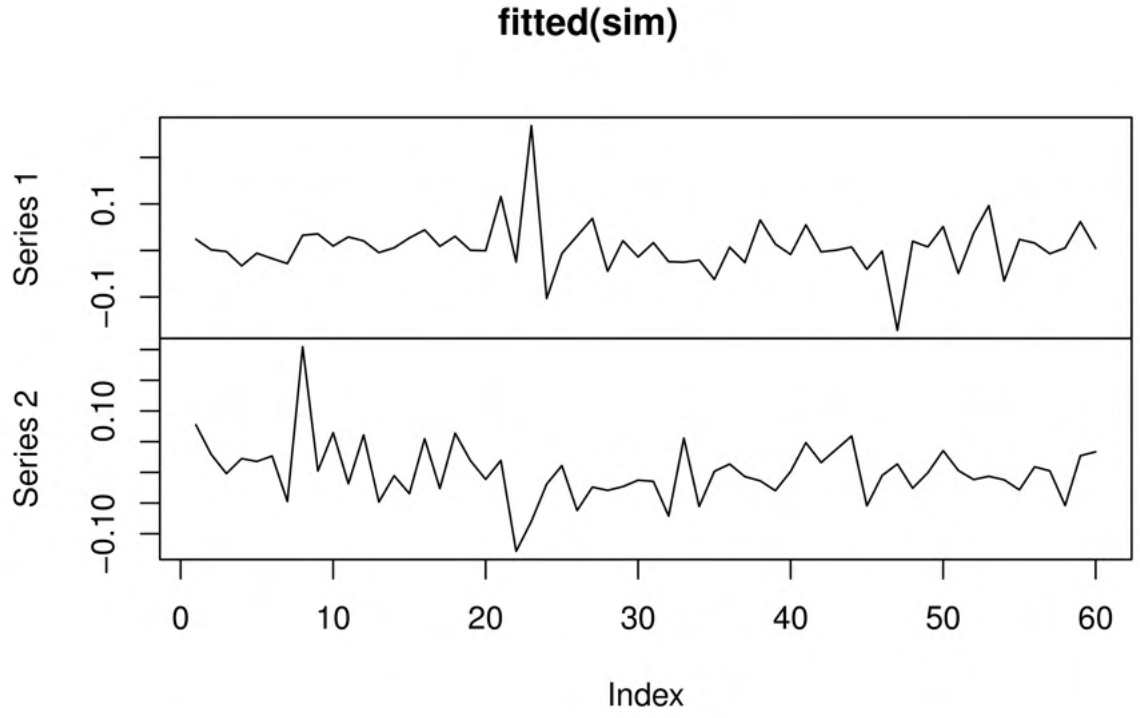

```
## Pearson goodness < 0.05: Normal is not a good model then

## impact of news:
## lets see whether the model is symmetric or not

news_norm = newsimpact(m_norm)
plot(news_norm$zx, news_norm$zy, ylab = news_norm$yexpr, xlab = news_norm$xexpr,main='News Impact')
```



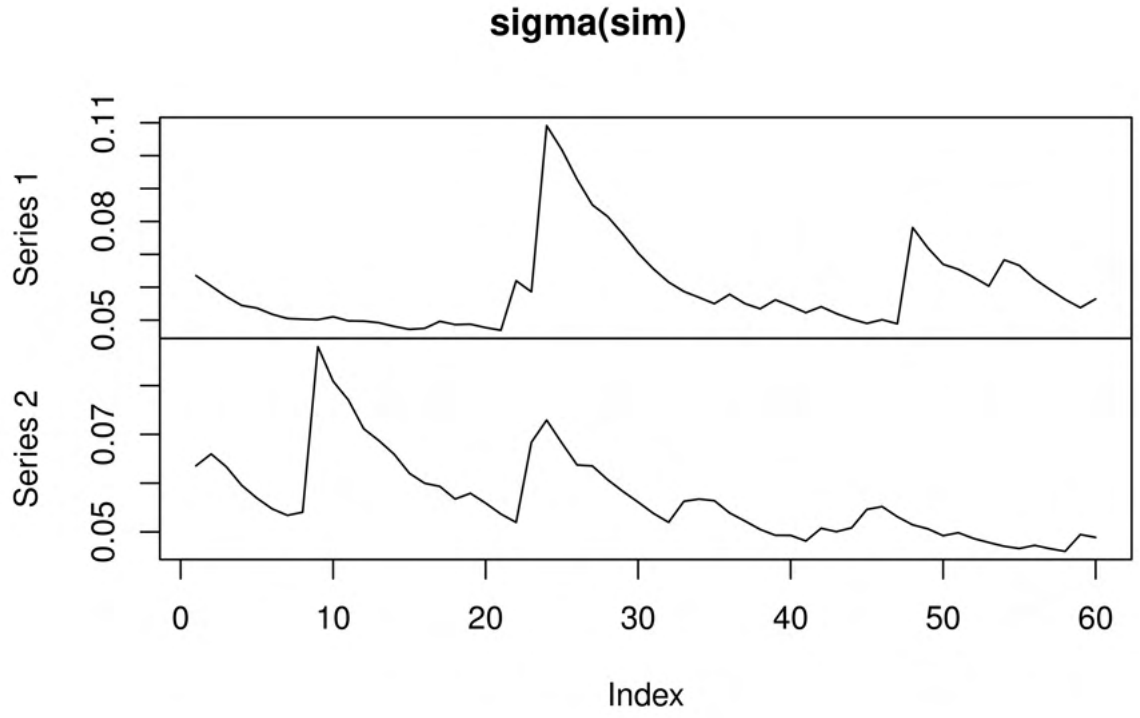

```
## symmetric Curve: positive and negative news have the same impact


## Volatility Forecast for next 60 days
sfinal = s_norm
setfixed(sfinal) = as.list(coef(m_norm))
s_norm_forecast = ugarchforecast(data = return,n.ahead = 60,fitORspec = sfinal)
plot(sigma(s_norm_forecast))
```




```
sim <- ugarchpath(spec = sfinal, m.sim = 2, n.sim = 1*60, rseed = 123)
plot.zoo(fitted(sim))
```



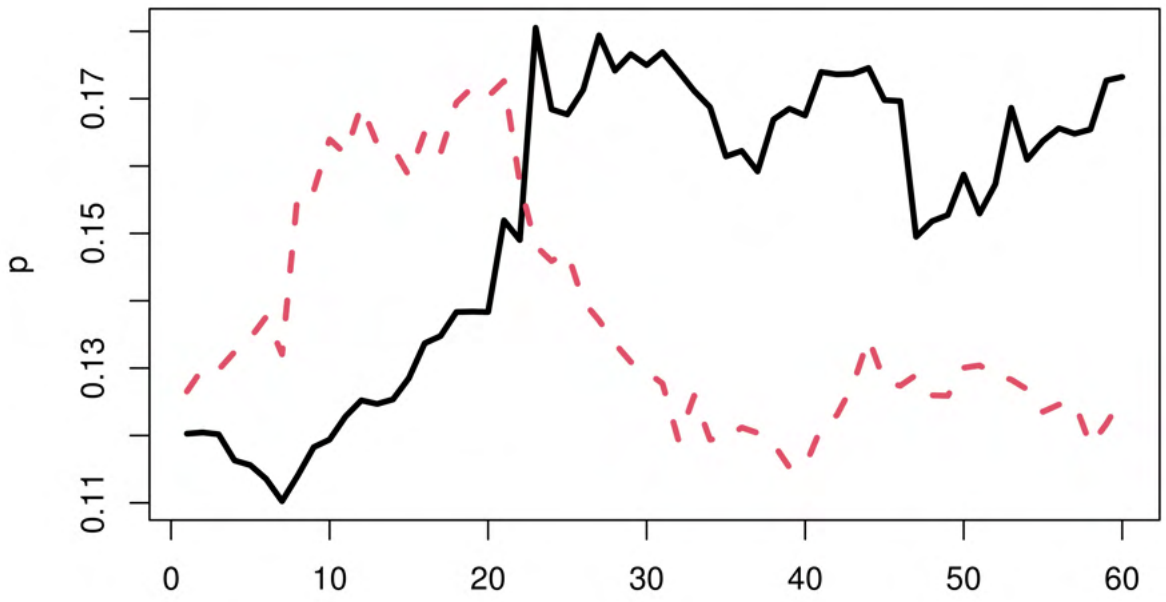

```
plot.zoo(sigma(sim))
```




```
print(tail(df$Close))
```

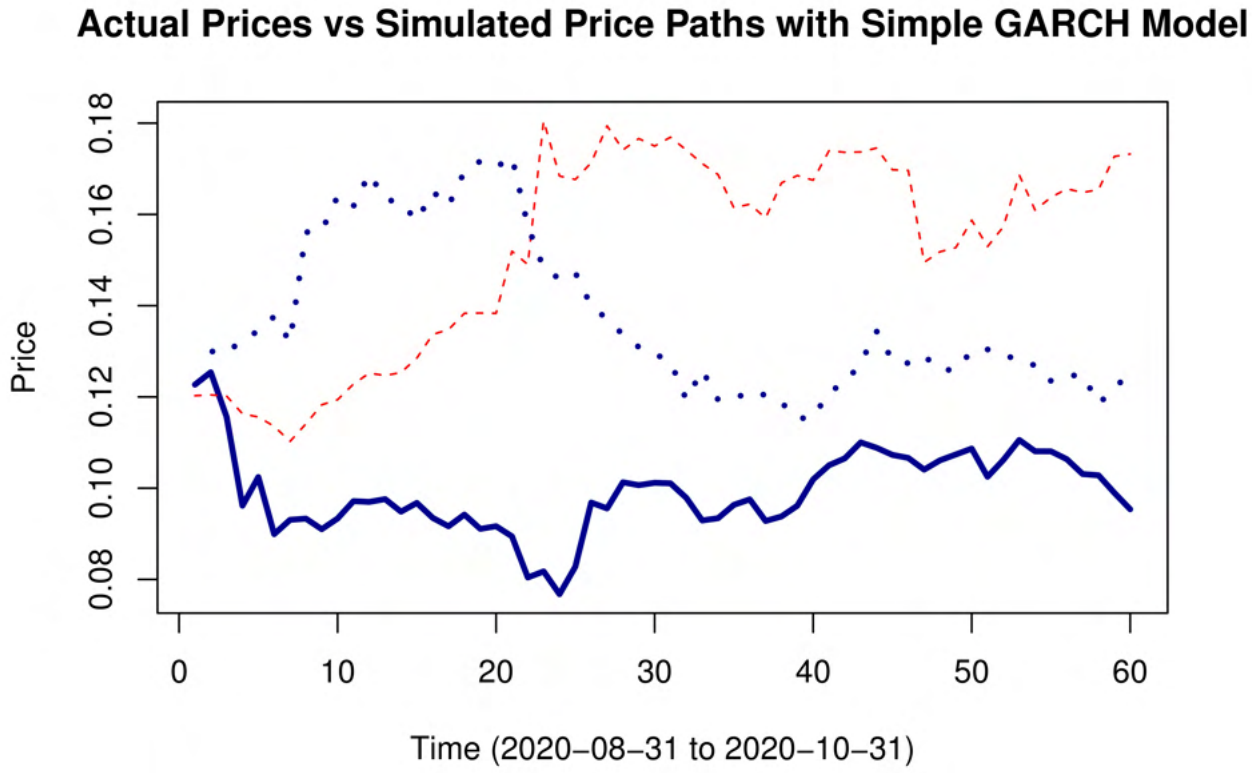
```
## [1] 0.1132921 0.1152852 0.1068243 0.1094526 0.1167117 0.1174374
```

```
last_value = tail(df$Close,1)

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

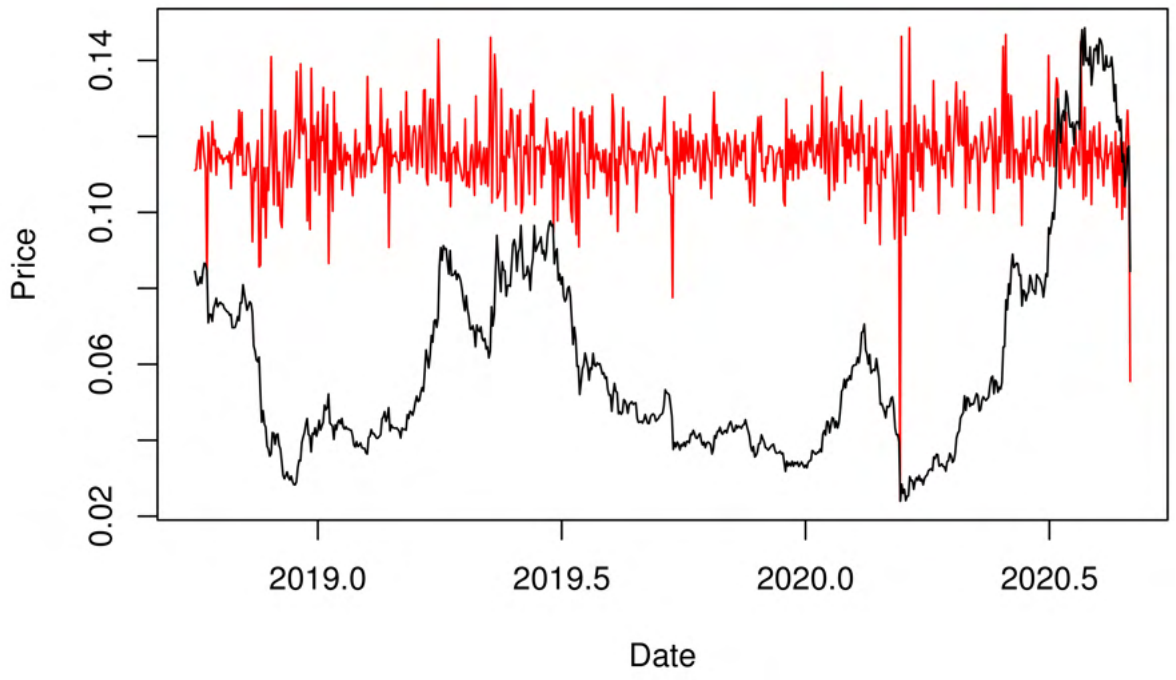



```
matplot(cbind(df2$Close, p), type = "l", lwd = c(3, 1), col = c("darkblue", "red"),
        xlab = paste("Time (", end_date, " to ", as.Date("2020-10-31"), ")", sep = ""),
        ylab = "Price", main = "Actual Prices vs Simulated Price Paths with Simple GARCH Model")
```

```
## Warning in cbind(df2$Close, p): number of rows of result is not a multiple of
## vector length (arg 1)
```




```
##############################################################################
############################################################

## Leverage effect is there: so apply the E-Garch Model
## gamma is the leverage variable
## if gamma positive: good news decreases the volatility

plot(return, type="l", col="red", xlab="", ylab="", axes=FALSE)
par(new=TRUE)
plot(df3$Close, type="l", xlab="Date", ylab="Price")
```



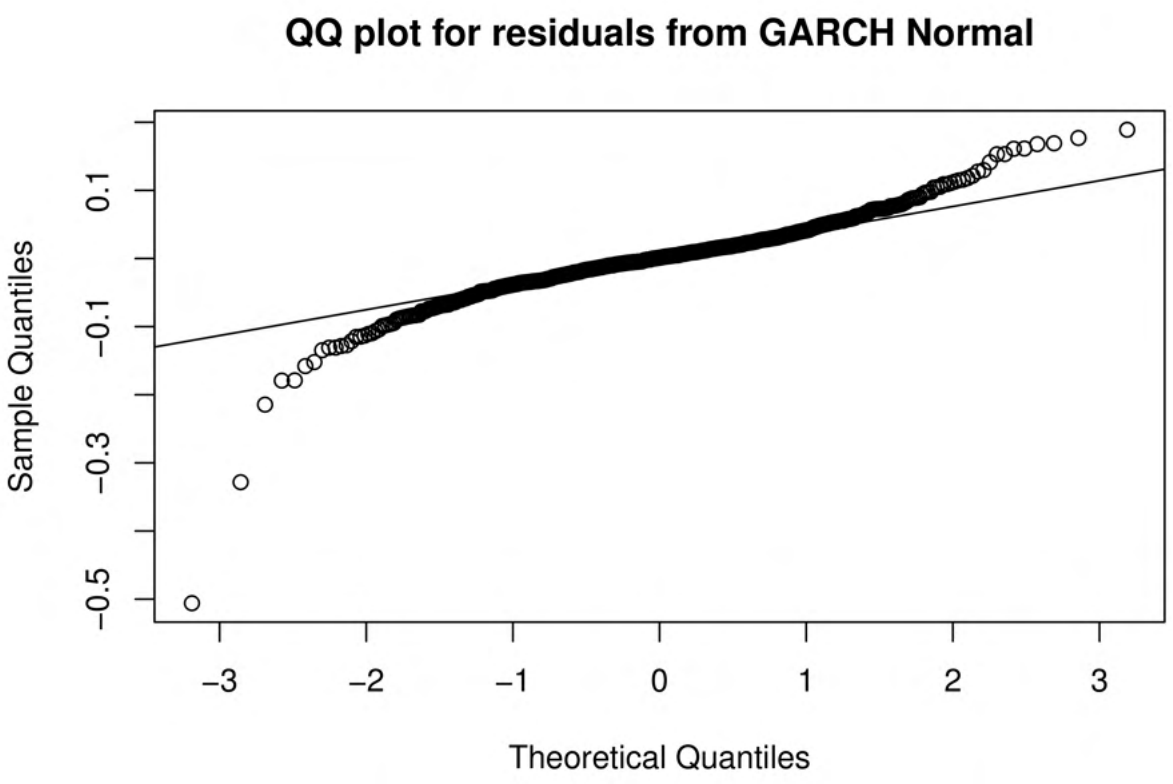

```
##########################################################################################
s_norm <- ugarchspec(mean.model = list(armaOrder = c(3,2)),variance.model = list(model = 'eGARCH',garch
                     distribution.model = 'sstd')

m_norm <- ugarchfit(data = return, spec = s_norm, solver ='hybrid')
m_norm
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(1,1)
## Mean Model   : ARFIMA(3,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000228    0.001046  -0.21826 0.827226
## ar1     0.384556    0.048580   7.91594 0.000000
## ar2     0.442573    0.068318   6.47816 0.000000
## ar3     0.030571    0.026709   1.14457 0.252386
## ma1    -0.510891    0.051054 -10.00691 0.000000
```

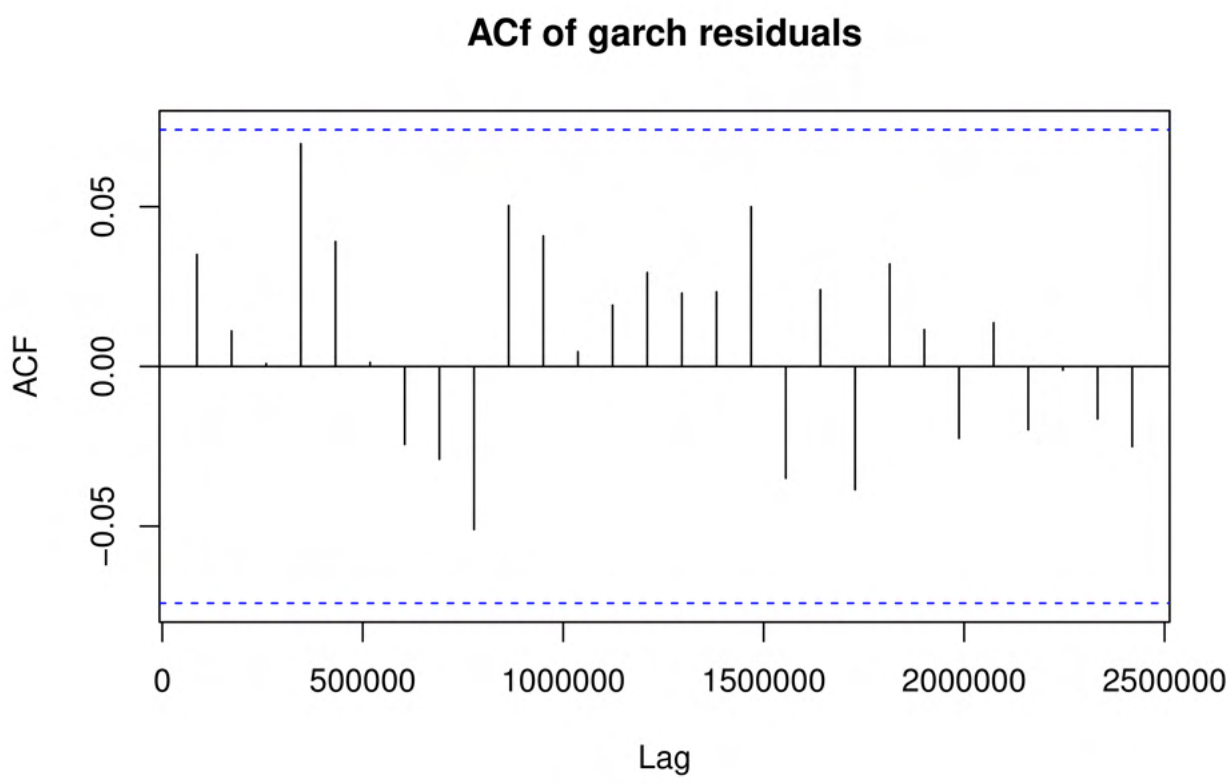
```
## ma2    -0.338453    0.062611  -5.40569 0.000000
## omega  -0.623208    0.315055  -1.97810 0.047918
## alpha1  0.016514    0.041401   0.39888 0.689982
## beta1   0.890580    0.054442  16.35817 0.000000
## gamma1  0.240069    0.077666   3.09104 0.001995
## skew    0.974663    0.041541  23.46282 0.000000
## shape   3.061490    0.422240   7.25059 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000228    0.000619  -0.36862 0.712409
## ar1     0.384556    0.014502  26.51675 0.000000
## ar2     0.442573    0.040427  10.94755 0.000000
## ar3     0.030571    0.019843   1.54061 0.123413
## ma1    -0.510891    0.019157 -26.66898 0.000000
## ma2    -0.338453    0.020096 -16.84152 0.000000
## omega  -0.623208    0.240774  -2.58836 0.009643
## alpha1  0.016514    0.035542   0.46462 0.642202
## beta1   0.890580    0.041139  21.64822 0.000000
## gamma1  0.240069    0.061985   3.87302 0.000107
## skew    0.974663    0.036449  26.74072 0.000000
## shape   3.061490    0.409005   7.48520 0.000000
##
## LogLikelihood : 1146.073
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.2402
## Bayes        -3.1622
## Shibata      -3.2408
## Hannan-Quinn -3.2101
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                             statistic p-value
## Lag[1]                          2.624  0.1053
## Lag[2*(p+q)+(p+q)-1][14]        6.793  0.8840
## Lag[4*(p+q)+(p+q)-1][24]       10.382  0.7818
## d.o.f=5
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                            statistic p-value
## Lag[1]                       0.07946  0.7780
## Lag[2*(p+q)+(p+q)-1][5]      0.94329  0.8724
## Lag[4*(p+q)+(p+q)-1][9]      1.82823  0.9245
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.09212 0.500 2.000  0.7615
```

```
## ARCH Lag[5]   1.45430 1.440 1.667  0.6044
## ARCH Lag[7]   1.88086 2.315 1.543  0.7425
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.6802
## Individual Statistics:
## mu     0.22876
## ar1    0.11357
## ar2    0.07487
## ar3    0.09899
## ma1    0.11306
## ma2    0.08123
## omega  0.08143
## alpha1 0.05822
## beta1  0.07974
## gamma1 0.17912
## skew   0.22231
## shape  0.09010
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.69 2.96 3.51
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.03231 0.9742
## Negative Sign Bias 0.17505 0.8611
## Positive Sign Bias 0.70845 0.4789
## Joint Effect       0.96946 0.8086
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     16.23       0.6420
## 2    30     35.60       0.1855
## 3    40     45.14       0.2307
## 4    50     37.43       0.8863
##
##
## Elapsed time : 0.620018
```

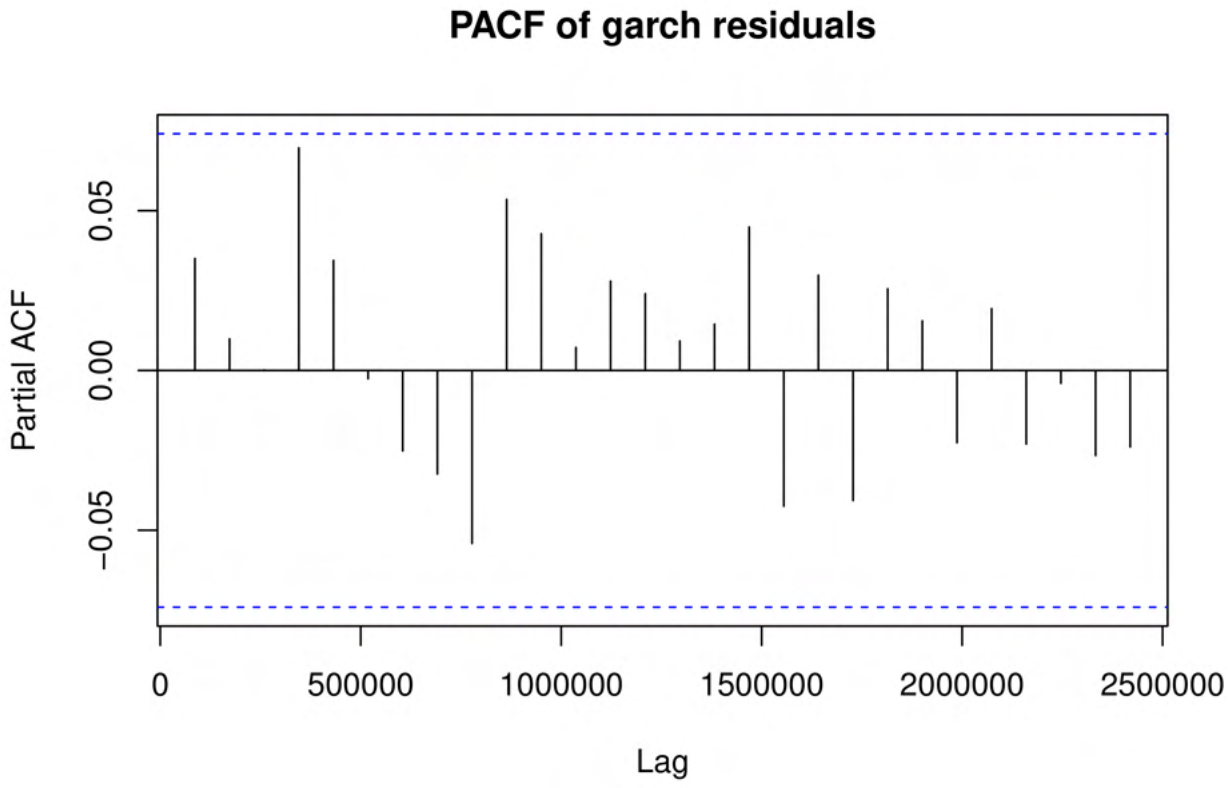
```
resid_norm = residuals(m_norm)
qqnorm(resid_norm,  main = 'QQ plot for residuals from GARCH Normal')
qqline(resid_norm)
```



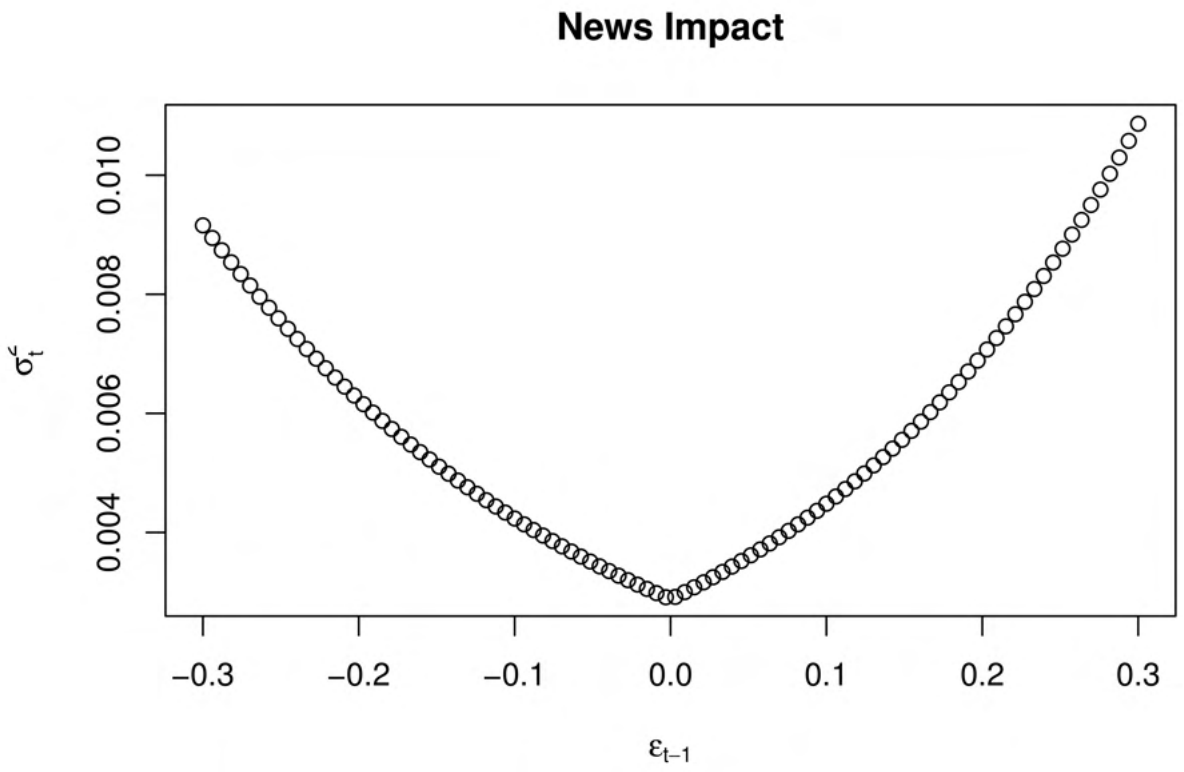

```
acf(resid_norm, main="ACf of garch residuals")
```



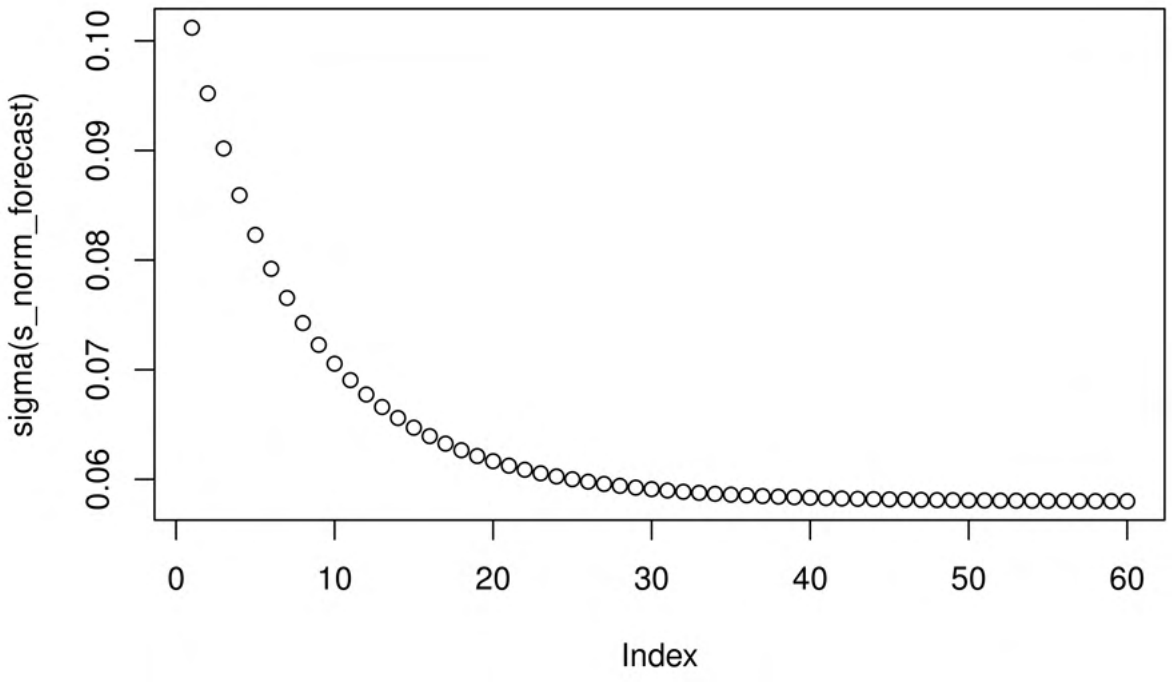

```
pacf(resid_norm, main = "PACF of garch residuals")
```



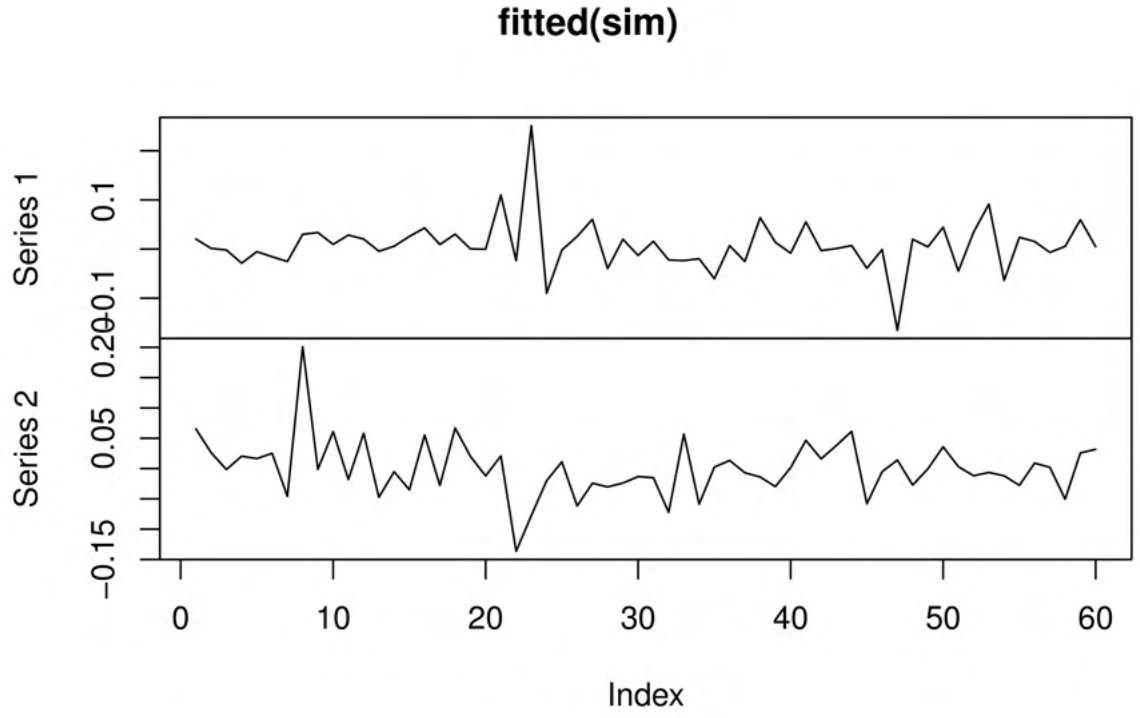

```
## Pearson goodness < 0.05: Normal is not a good model then

## impact of news:
## lets see whether the model is symmetric or not

news_norm = newsimpact(m_norm)
plot(news_norm$zx, news_norm$zy, ylab = news_norm$yexpr, xlab = news_norm$xexpr,main='News Impact')
```



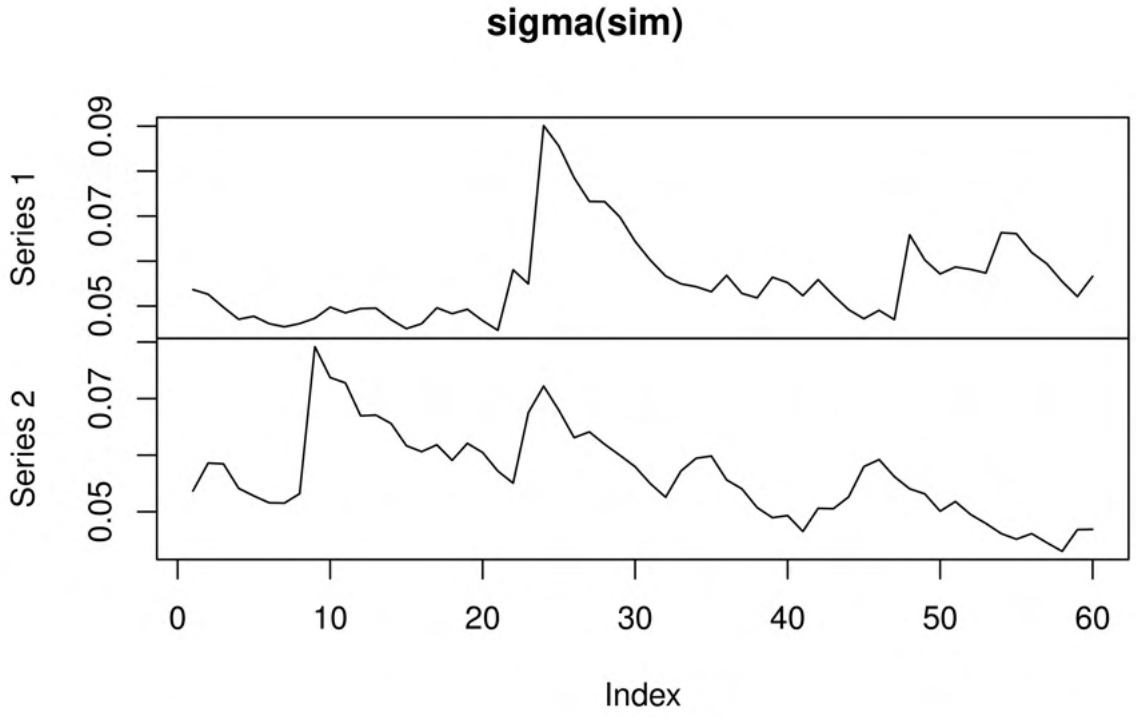

```
## symmetric Curve: positive and negative news have the same impact

## Volatility Forecast for next 2 0 days
sfinal = s_norm
setfixed(sfinal) = as.list(coef(m_norm))
s_norm_forecast = ugarchforecast(data = return,n.ahead = 60,fitORspec = sfinal)
plot(sigma(s_norm_forecast))
```




```
sim <- ugarchpath(spec = sfinal, m.sim = 2, n.sim = 1*60, rseed = 123)
plot.zoo(fitted(sim))
```



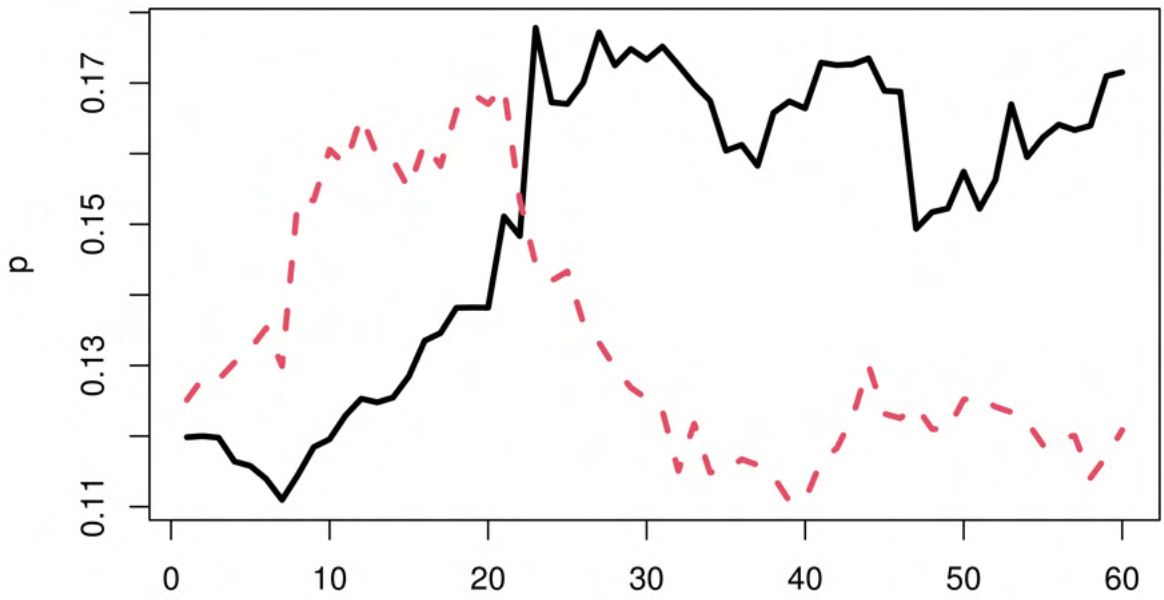

```
plot.zoo(sigma(sim))
```




```
print(tail(df$Close))
```

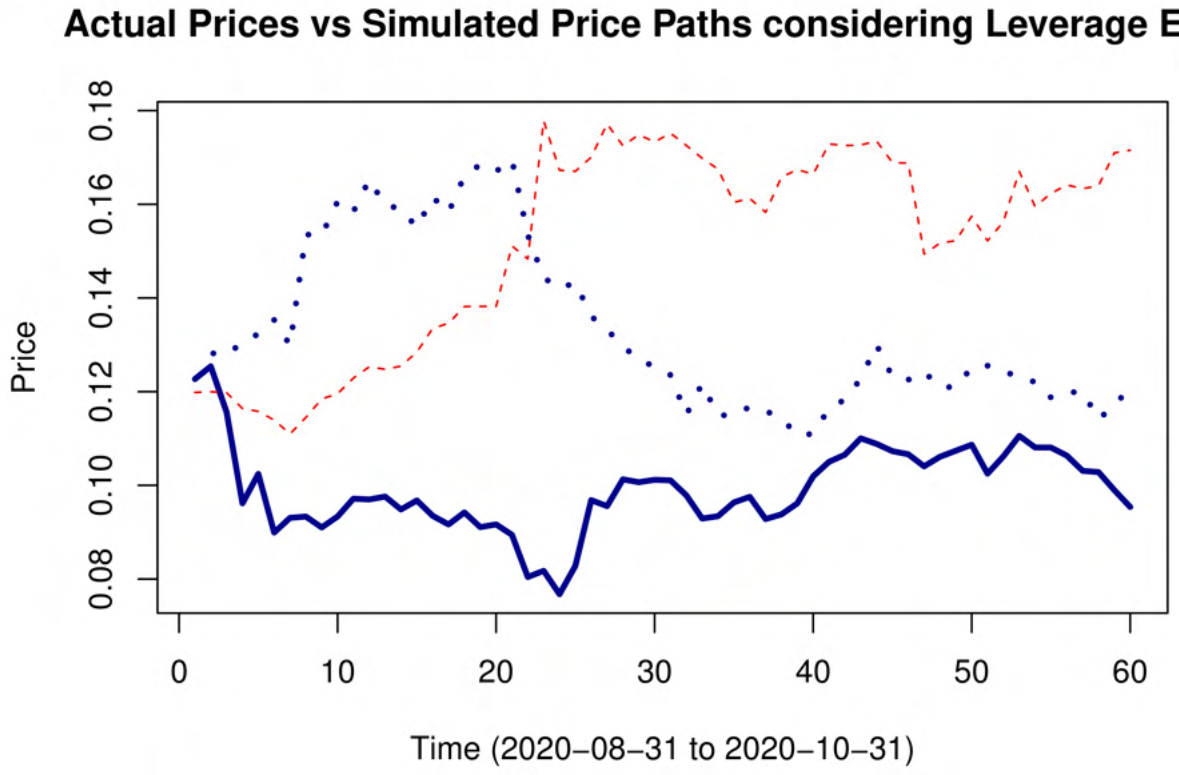
```
## [1] 0.1132921 0.1152852 0.1068243 0.1094526 0.1167117 0.1174374
```

```
last_value = tail(df$Close,1)

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




```
matplot(cbind(df2$Close, p), type = "l", lwd = c(3, 1), col = c("darkblue", "red"),
        xlab = paste("Time (", end_date, " to ", as.Date("2020-10-31"), ")", sep = ""),
        ylab = "Price", main = "Actual Prices vs Simulated Price Paths considering Leverage Effect")
```

```
## Warning in cbind(df2$Close, p): number of rows of result is not a multiple of
## vector length (arg 1)
```

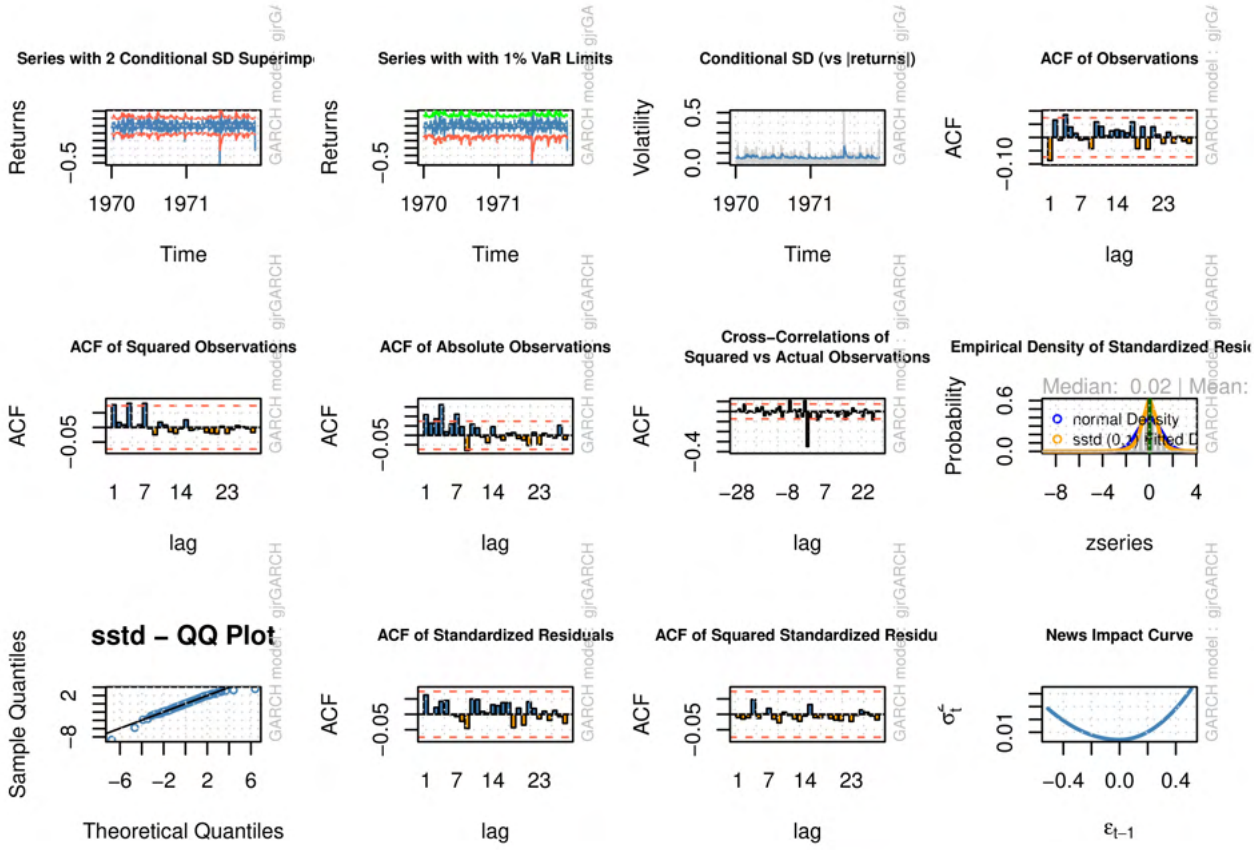



```
#########################################################################################
#########################################################################################

return = (diff(log(df3$Close)))

s_norm <- ugarchspec(mean.model = list(armaOrder = c(3,2)),variance.model = list(model = 'gjrGARCH',gar
                     distribution.model = 'sstd')

m_norm <- ugarchfit(data = return, spec = s_norm, solver ='hybrid')
m_norm
```

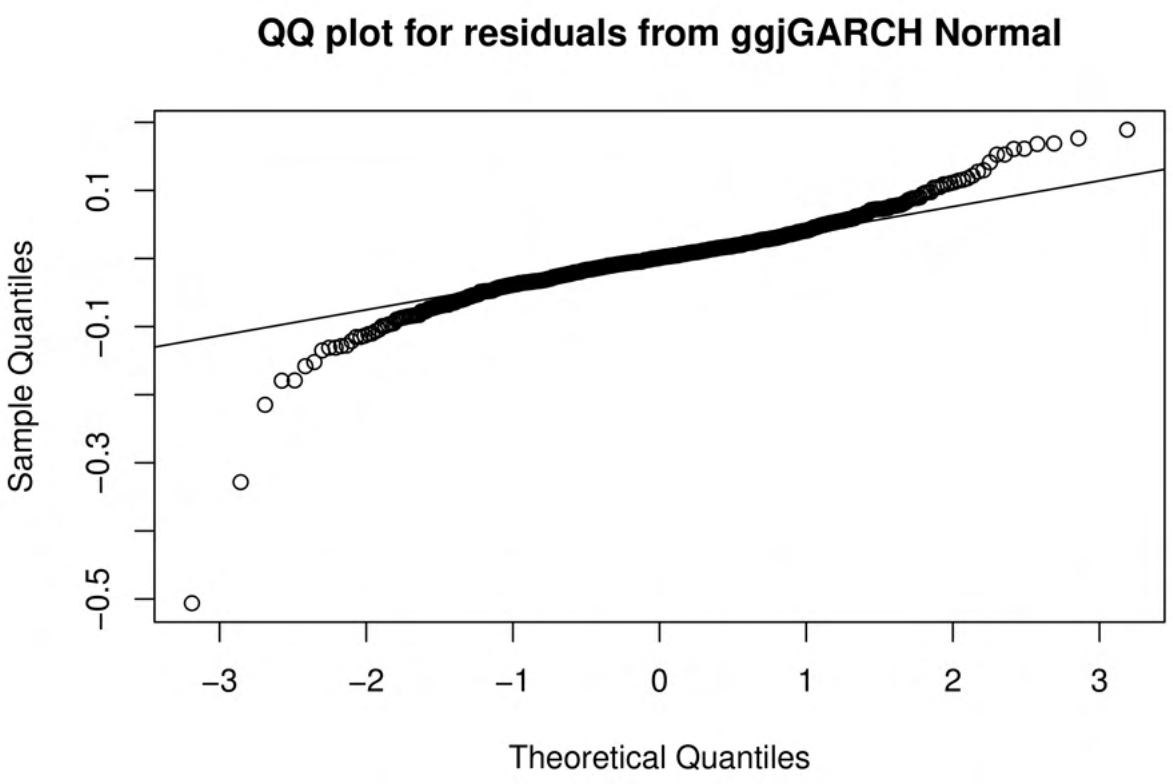
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(3,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000045    0.002032 -0.022337 0.982179
```

```
## ar1     0.375339    0.394653  0.951062 0.341573
## ar2     0.447581    0.298878  1.497538 0.134253
## ar3     0.034598    0.064820  0.533758 0.593509
## ma1    -0.502071    0.393382 -1.276294 0.201852
## ma2    -0.347480    0.348737 -0.996396 0.319058
## omega   0.000405    0.000186  2.177804 0.029421
## alpha1  0.152348    0.070091  2.173580 0.029737
## beta1   0.790889    0.069959 11.305024 0.000000
## gamma1 -0.058789    0.067396 -0.872291 0.383050
## skew    0.975428    0.049405 19.743454 0.000000
## shape   3.030699    0.413115  7.336205 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000045    0.002240 -0.020257 0.983838
## ar1     0.375339    0.107005  3.507693 0.000452
## ar2     0.447581    0.102094  4.384007 0.000012
## ar3     0.034598    0.030061  1.150918 0.249766
## ma1    -0.502071    0.105558 -4.756359 0.000002
## ma2    -0.347480    0.106885 -3.250986 0.001150
## omega   0.000405    0.000129  3.133640 0.001727
## alpha1  0.152348    0.052214  2.917748 0.003526
## beta1   0.790889    0.049349 16.026514 0.000000
## gamma1 -0.058789    0.063596 -0.924414 0.355271
## skew    0.975428    0.052148 18.704971 0.000000
## shape   3.030699    0.401407  7.550184 0.000000
##
## LogLikelihood : 1145.817
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.2395
## Bayes        -3.1615
## Shibata      -3.2401
## Hannan-Quinn -3.2093
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.778 0.09555
## Lag[2*(p+q)+(p+q)-1][14]    7.079 0.75550
## Lag[4*(p+q)+(p+q)-1][24]   10.918 0.70988
## d.o.f=5
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                        statistic p-value
## Lag[1]                   0.07931  0.7782
## Lag[2*(p+q)+(p+q)-1][5]  0.93261  0.8747
## Lag[4*(p+q)+(p+q)-1][9]  1.61975  0.9452
## d.o.f=2
##
```

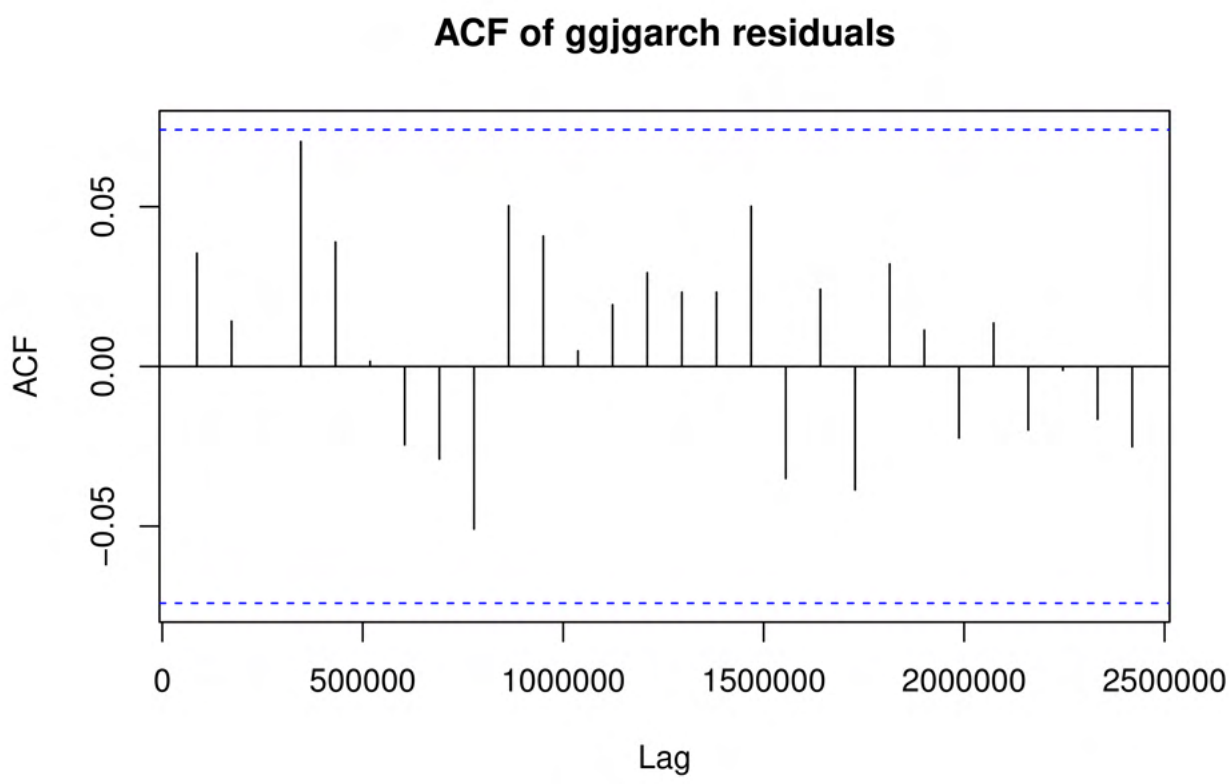
```
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.08496 0.500 2.000  0.7707
## ARCH Lag[5]   1.44017 1.440 1.667  0.6082
## ARCH Lag[7]   1.65548 2.315 1.543  0.7897
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.8188
## Individual Statistics:
## mu     0.25568
## ar1    0.14087
## ar2    0.07705
## ar3    0.11093
## ma1    0.13824
## ma2    0.08519
## omega  0.08382
## alpha1 0.18502
## beta1  0.10255
## gamma1 0.10182
## skew   0.22207
## shape  0.10426
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.69 2.96 3.51
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.03581 0.9714
## Negative Sign Bias 0.27053 0.7868
## Positive Sign Bias 0.71192 0.4768
## Joint Effect       0.97928 0.8063
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     13.43       0.8159
## 2    30     34.83       0.2102
## 3    40     33.26       0.7286
## 4    50     40.14       0.8124
##
##
## Elapsed time : 1.428219
plot(m_norm, which='all')

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



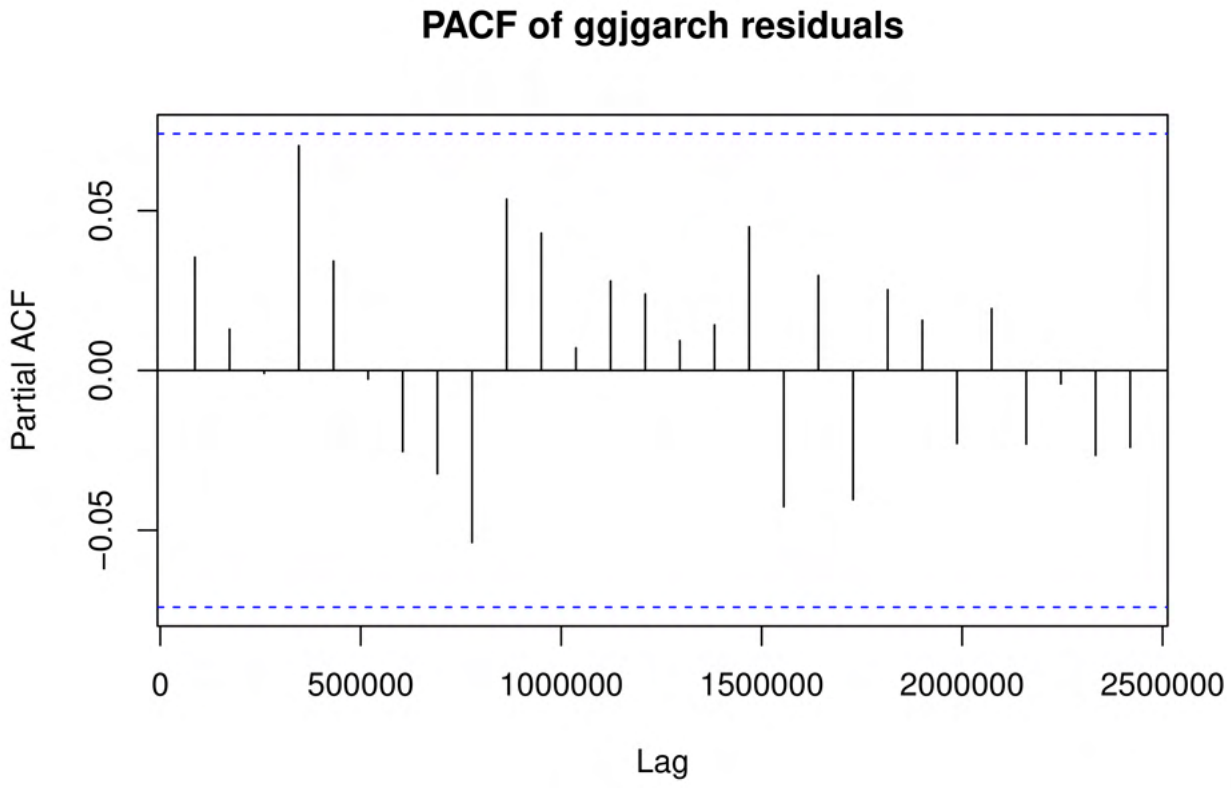

```
resid_norm = residuals(m_norm)
qqnorm(resid_norm,  main = 'QQ plot for residuals from ggjGARCH Normal')
qqline(resid_norm)
```



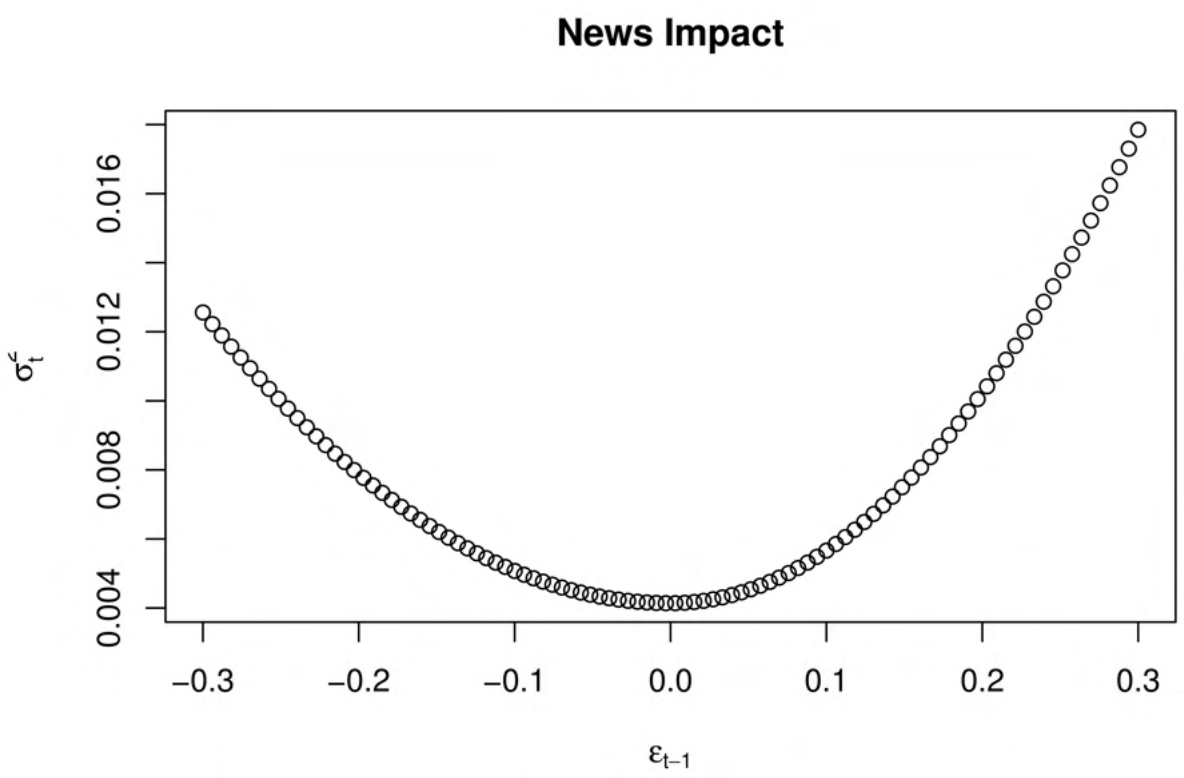

```
acf(resid_norm, main="ACF of ggjgarch residuals")
```



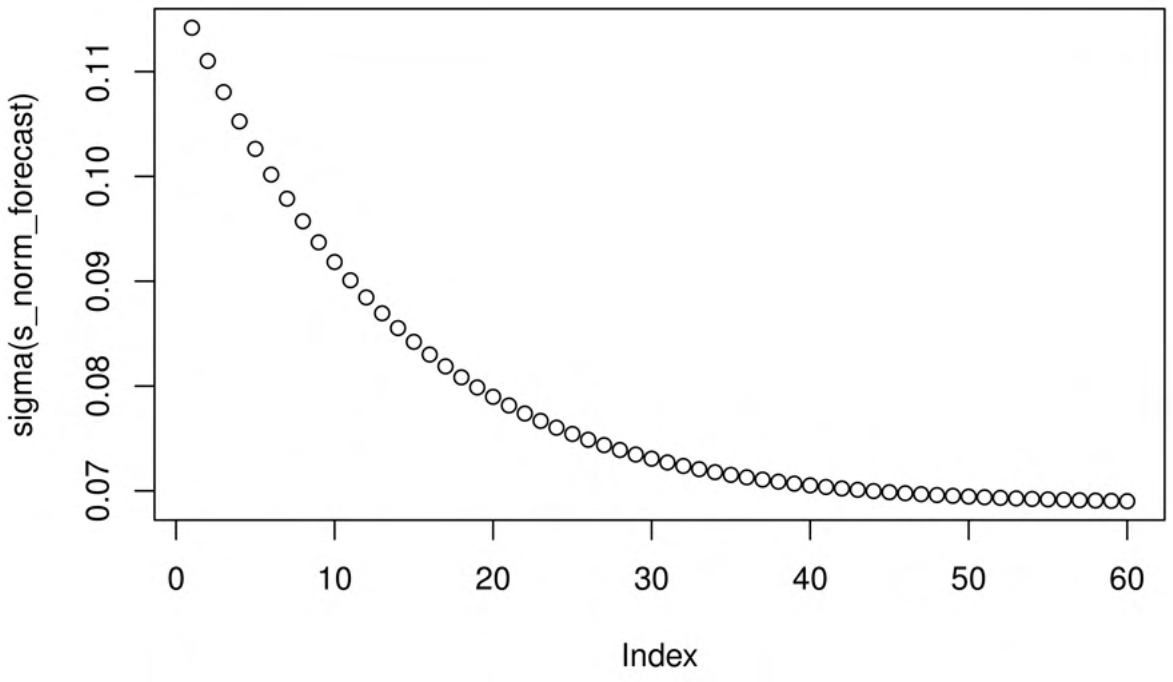

```
pacf(resid_norm, main = "PACF of ggjgarch residuals")
```



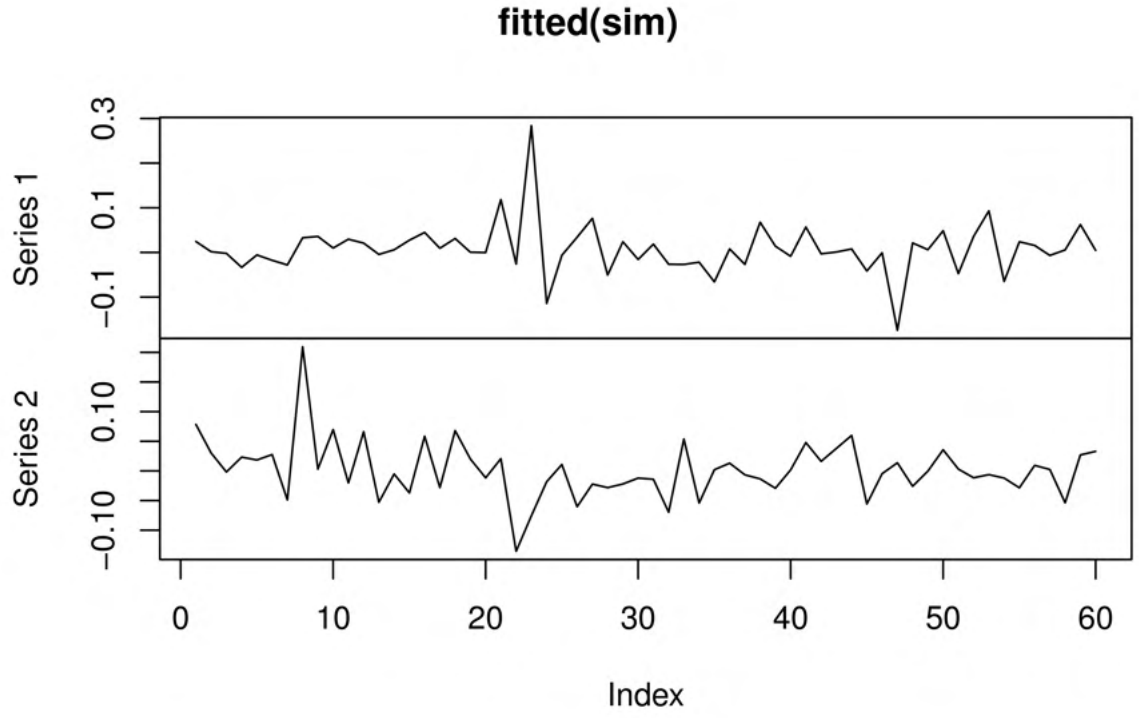

```
## Pearson goodness < 0.05: Normal is not a good model then

## impact of news:
## lets see whether the model is symmetric or not

news_norm = newsimpact(m_norm)
plot(news_norm$zx, news_norm$zy, ylab = news_norm$yexpr, xlab = news_norm$xexpr,main='News Impact')
```



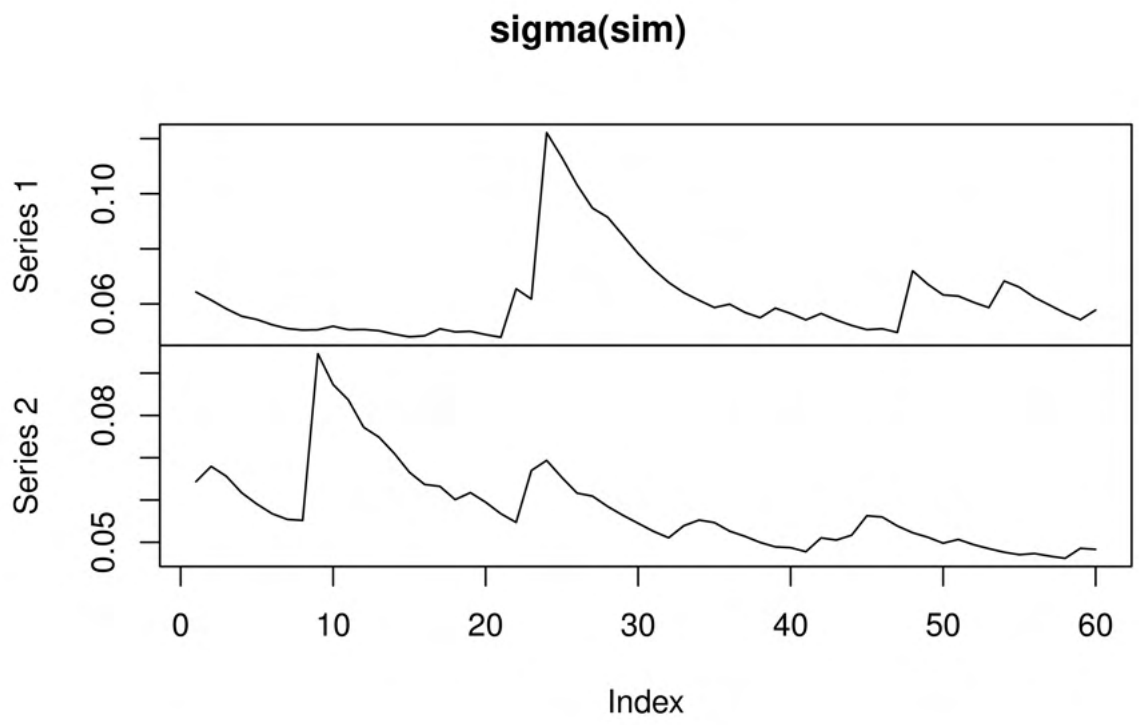

```
## symmetric Curve: positive and negative news have the same impact

## Volatility Forecast for next 2 0 days
sfinal = s_norm
setfixed(sfinal) = as.list(coef(m_norm))
s_norm_forecast = ugarchforecast(data = return,n.ahead = 60,fitORspec = sfinal)
plot(sigma(s_norm_forecast))
```




```
sim <- ugarchpath(spec = sfinal, m.sim = 2, n.sim = 1*60, rseed = 123)
plot.zoo(fitted(sim))
```




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



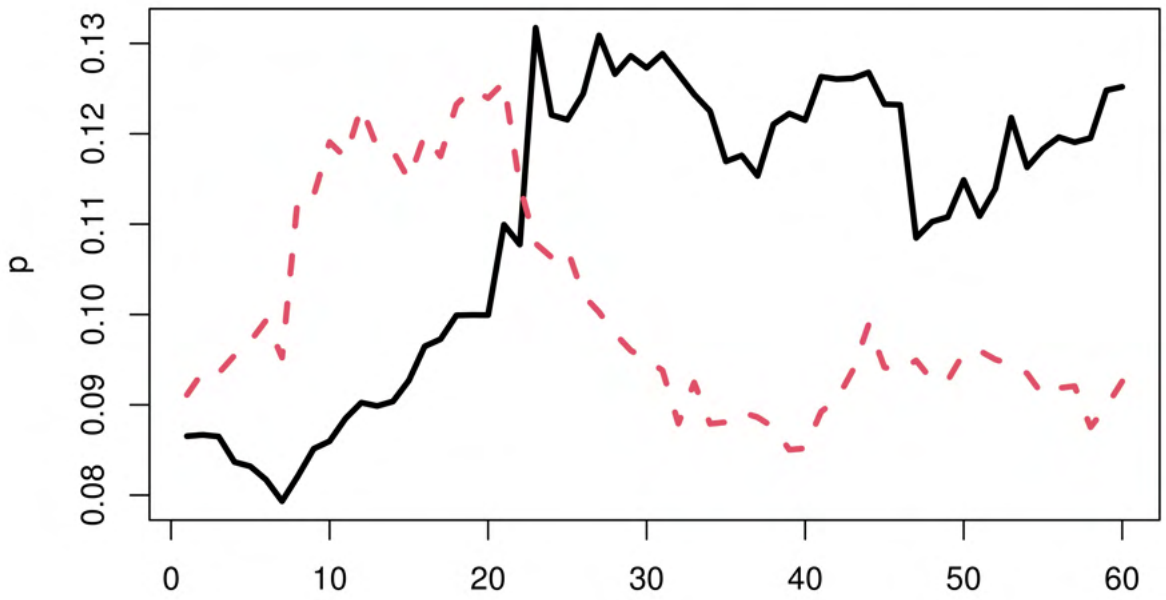

```
print(tail(df3$Close))
```

```
## Time Series:
## Start = c(2020, 239)
## End = c(2020, 244)
## Frequency = 365
## [1] 0.1152852 0.1068243 0.1094526 0.1167117 0.1174374 0.0844693
```

```
last_value = tail(df3$Close,1)[1]
last_value
```

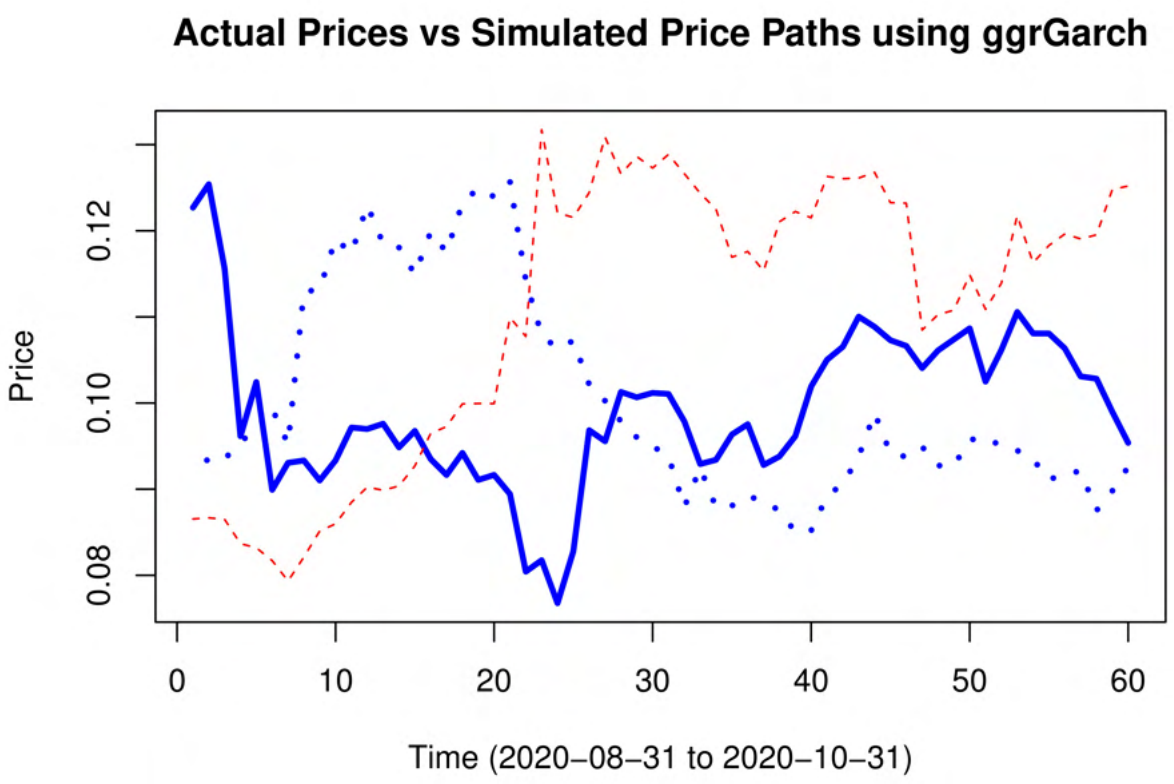
```
## [1] 0.0844693
```

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




```
matplot(cbind(df2$Close, p), type = "l", lwd = c(3, 1), col = c("blue", "red"),
        xlab = paste("Time (", end_date, " to ", as.Date("2020-10-31"), ")", sep = ""),
        ylab = "Price", main = "Actual Prices vs Simulated Price Paths using ggrGarch")
```

```
## Warning in cbind(df2$Close, p): number of rows of result is not a multiple of
## vector length (arg 1)
```


Actual Prices vs Simulated Price Paths using ggrGarch
Price
0.08
0.10
0.12
0
10
20
30
40
50
60
Time (2020−08−31 to 2020−10−31)