INDONESIAN PARADISE PROPERTY

2023-07-28

# Load the required libraries  
library(readxl)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(forecast)

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

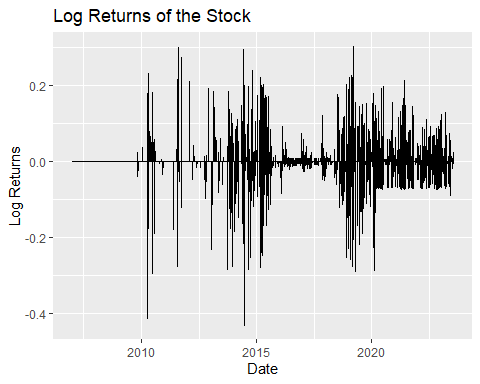
library(rugarch)

## Loading required package: parallel

##   
## Attaching package: 'rugarch'

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

library(tseries)  
  
# Read the Excel data  
data <- read\_excel("C:/Users/fredy/Desktop/INDONESIAN PARADISE PROPERTY.xlsx")  
# Replace "null" with NA in numeric columns  
numeric\_columns <- c("Open", "High", "Low", "Close", "Adj Close", "Volume")  
data[, numeric\_columns][data[, numeric\_columns] == "null"] <- NA  
data[, numeric\_columns] <- apply(data[, numeric\_columns], 2, as.numeric)  
  
# Convert the "Date" column to the correct date format  
data$Date <- as.Date(data$Date, format = "%m/%d/%Y")  
# Calculate log returns of the stock  
data$log\_returns <- log(data$Close) - log(lag(data$Close))  
# Remove the first row (NAs due to the lag operation)  
data <- na.omit(data)  
# Plot the log returns to visualize the time series  
ggplot(data, aes(x = Date, y = log\_returns)) +  
 geom\_line() +  
 labs(title = "Log Returns of the Stock", x = "Date", y = "Log Returns")



# Perform the Augmented Dickey-Fuller (ADF) test  
adf\_test <- adf.test(data$log\_returns, alternative = "stationary")

## Warning in adf.test(data$log\_returns, alternative = "stationary"): p-value  
## smaller than printed p-value

#Extract the p-value from the test results  
p\_value <- adf\_test$p.value  
# Print the p-value  
cat("ADF Test p-value:", p\_value, "\n")

## ADF Test p-value: 0.01

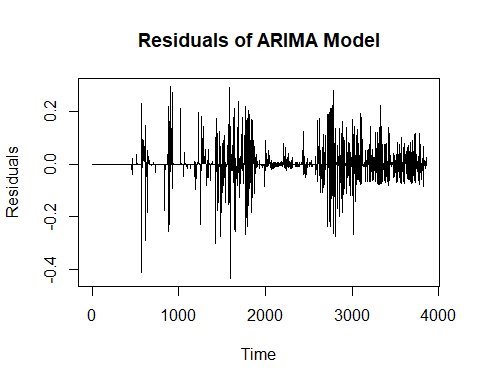
# Check if the time series is stationary based on the p-value  
if (p\_value < 0.05) {  
 print("The log returns time series is stationary.")  
} else {  
 print("The log returns time series is not stationary.")  
}

## [1] "The log returns time series is stationary."

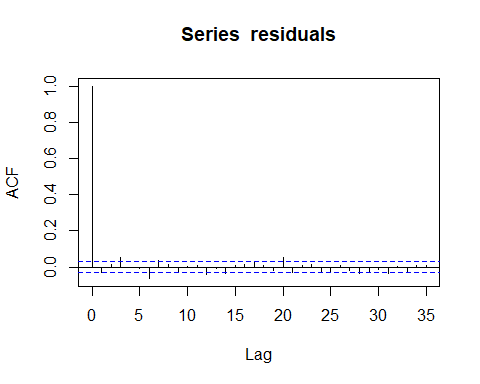
#The data is stationary, therefore, we continue modelling the arima  
arima\_model <- Arima(data$log\_returns, order = c(1, 0, 1))  
# Print the ARIMA model summary  
summary(arima\_model)

## Series: data$log\_returns   
## ARIMA(1,0,1) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 mean  
## 0.6799 -0.8066 3e-04  
## s.e. 0.0535 0.0438 4e-04  
##   
## sigma^2 = 0.001785: log likelihood = 6727.55  
## AIC=-13447.1 AICc=-13447.09 BIC=-13422.07  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -1.341401e-06 0.04223312 0.01740828 NaN Inf 0.6463645 -0.02836247

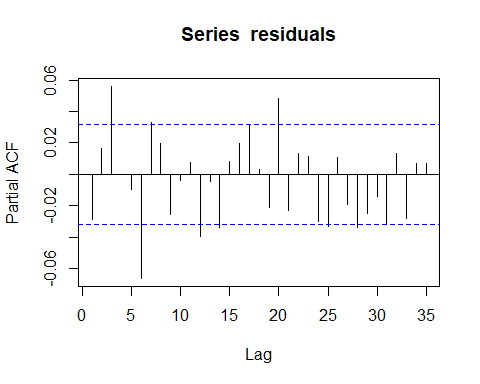
# Extract the residuals from the ARIMA model  
residuals <- residuals(arima\_model)  
# Plot the residuals to check for any patterns or trends  
plot(residuals, main = "Residuals of ARIMA Model", ylab = "Residuals")



# Plot the ACF (Autocorrelation Function) of the residuals  
acf(residuals)



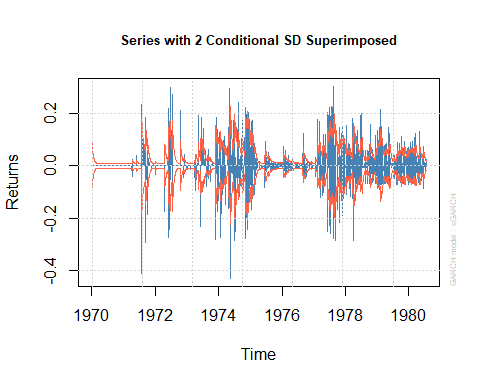
# Plot the PACF (Partial Autocorrelation Function) of the residuals  
pacf(residuals)



# Fit the GARCH(1,1) model  
garch\_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)))  
garch\_model <- ugarchfit(spec = garch\_spec, data = data$log\_returns)  
# Print the GARCH model summary  
print(garch\_model)

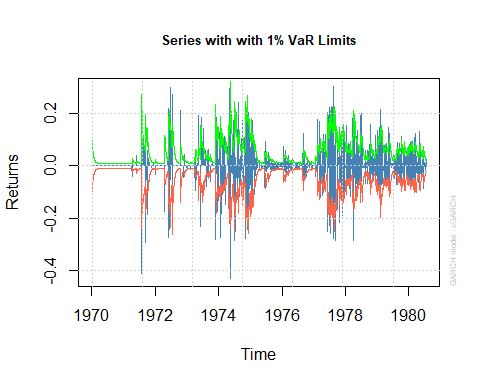
##   
## \*---------------------------------\*  
## \* GARCH Model Fit \*  
## \*---------------------------------\*  
##   
## Conditional Variance Dynamics   
## -----------------------------------  
## GARCH Model : sGARCH(1,1)  
## Mean Model : ARFIMA(1,0,1)  
## Distribution : norm   
##   
## Optimal Parameters  
## ------------------------------------  
## Estimate Std. Error t value Pr(>|t|)  
## mu -0.000455 0.000000 -3958.1349 0  
## ar1 0.675192 0.000170 3976.2179 0  
## ma1 -0.809044 0.000202 -3997.0994 0  
## omega 0.000002 0.000000 9.1336 0  
## alpha1 0.050757 0.000013 3903.3939 0  
## beta1 0.902710 0.000198 4549.0379 0  
##   
## Robust Standard Errors:  
## Estimate Std. Error t value Pr(>|t|)  
## mu -0.000455 0.004475 -0.101570 0.91910  
## ar1 0.675192 6.568559 0.102791 0.91813  
## ma1 -0.809044 7.768590 -0.104143 0.91706  
## omega 0.000002 0.014232 0.000131 0.99990  
## alpha1 0.050757 0.515748 0.098414 0.92160  
## beta1 0.902710 4.063295 0.222162 0.82419  
##   
## LogLikelihood : 7989.783   
##   
## Information Criteria  
## ------------------------------------  
##   
## Akaike -4.1431  
## Bayes -4.1334  
## Shibata -4.1431  
## Hannan-Quinn -4.1397  
##   
## Weighted Ljung-Box Test on Standardized Residuals  
## ------------------------------------  
## statistic p-value  
## Lag[1] 0.7004 0.40266  
## Lag[2\*(p+q)+(p+q)-1][5] 4.0621 0.05676  
## Lag[4\*(p+q)+(p+q)-1][9] 5.1385 0.42269  
## d.o.f=2  
## H0 : No serial correlation  
##   
## Weighted Ljung-Box Test on Standardized Squared Residuals  
## ------------------------------------  
## statistic p-value  
## Lag[1] 0.006163 0.9374  
## Lag[2\*(p+q)+(p+q)-1][5] 0.015365 1.0000  
## Lag[4\*(p+q)+(p+q)-1][9] 0.023915 1.0000  
## d.o.f=2  
##   
## Weighted ARCH LM Tests  
## ------------------------------------  
## Statistic Shape Scale P-Value  
## ARCH Lag[3] 1.474e-05 0.500 2.000 0.9969  
## ARCH Lag[5] 9.218e-03 1.440 1.667 0.9996  
## ARCH Lag[7] 1.388e-02 2.315 1.543 1.0000  
##   
## Nyblom stability test  
## ------------------------------------  
## Joint Statistic: 19.0702  
## Individual Statistics:   
## mu 0.7784  
## ar1 1.3306  
## ma1 1.3915  
## omega 0.1273  
## alpha1 9.4688  
## beta1 0.6366  
##   
## Asymptotic Critical Values (10% 5% 1%)  
## Joint Statistic: 1.49 1.68 2.12  
## Individual Statistic: 0.35 0.47 0.75  
##   
## Sign Bias Test  
## ------------------------------------  
## t-value prob sig  
## Sign Bias 1.05959 0.2894   
## Negative Sign Bias 0.02325 0.9815   
## Positive Sign Bias 0.72398 0.4691   
## Joint Effect 1.58059 0.6638   
##   
##   
## Adjusted Pearson Goodness-of-Fit Test:  
## ------------------------------------  
## group statistic p-value(g-1)  
## 1 20 7645 0  
## 2 30 7588 0  
## 3 40 11773 0  
## 4 50 11637 0  
##   
##   
## Elapsed time : 0.9475372

# Plot the GARCH model diagnostics  
plot(garch\_model, which = 1)

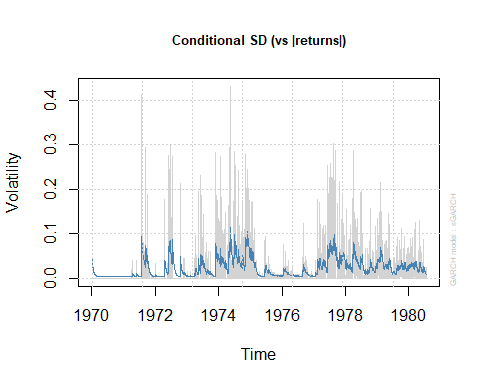


plot(garch\_model, which = 2)

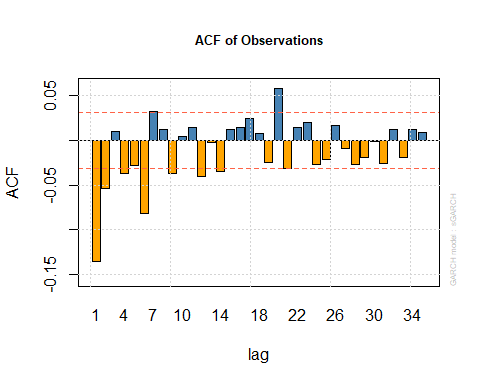
##   
## please wait...calculating quantiles...



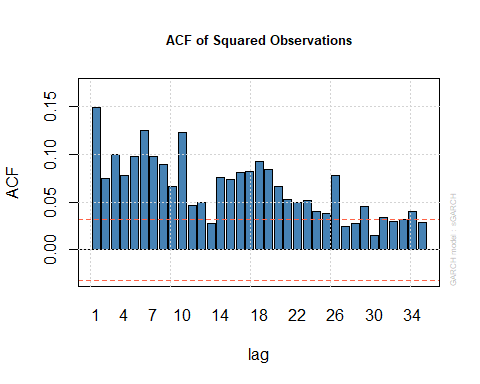
plot(garch\_model, which = 3)



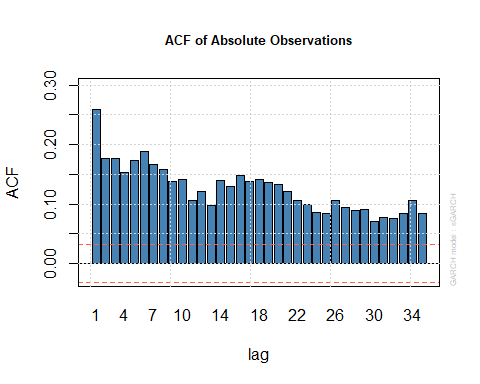
plot(garch\_model, which = 4)



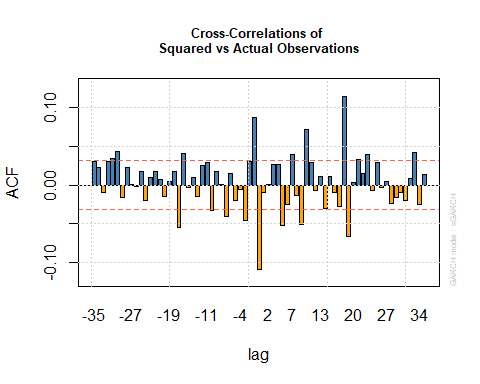
plot(garch\_model, which = 5)



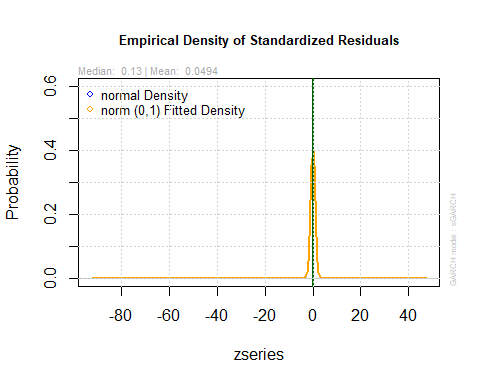
plot(garch\_model, which = 6)



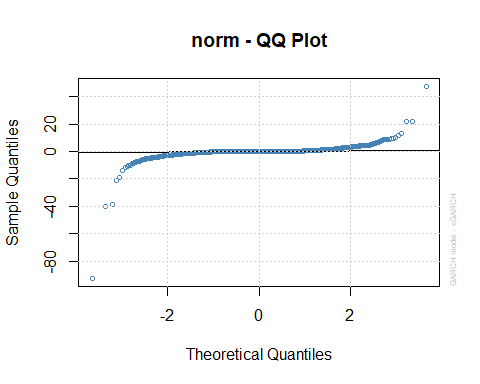
plot(garch\_model, which = 7)



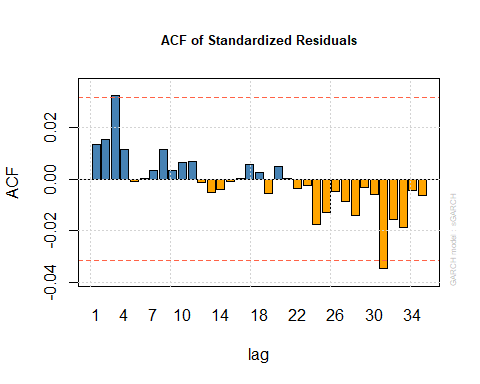
plot(garch\_model, which = 8)



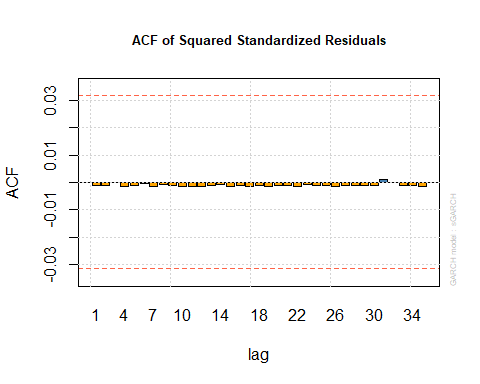
plot(garch\_model, which = 9)



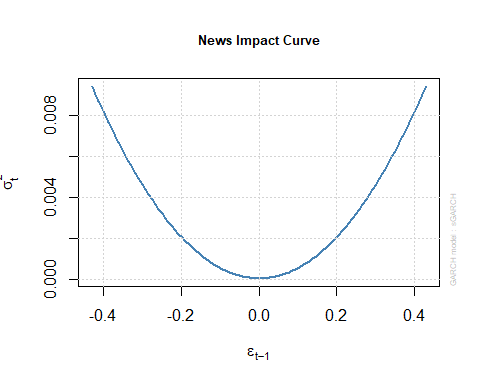
plot(garch\_model, which = 10)



plot(garch\_model, which = 11)



plot(garch\_model, which = 12)



2# Fit sGARCH model

## [1] 2

sgarch\_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)))  
sgarch\_model <- ugarchfit(spec = sgarch\_spec, data = data$log\_returns)  
  
# Fit sGARCH-sstd model  
sgarch\_sstd\_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),  
 mean.model = list(armaOrder = c(0, 0), include.mean = TRUE, archm = TRUE))  
sgarch\_sstd\_model <- ugarchfit(spec = sgarch\_sstd\_spec, data = data$log\_returns)  
  
# Fit GJR-GARCH model  
gjrgarch\_spec <- ugarchspec(variance.model = list(model = "gjrGARCH", garchOrder = c(1, 1, 1)))  
gjrgarch\_model <- ugarchfit(spec = gjrgarch\_spec, data = data$log\_returns)  
  
# Fit TGARCH model  
tgarch\_spec <- ugarchspec(variance.model = list(model = "iGARCH", garchOrder = c(1, 1), submodel = NULL, external.regressors = NULL))  
tgarch\_model <- ugarchfit(spec = tgarch\_spec, data = data$log\_returns)  
  
# Function to get AIC and BIC for uGARCHfit objects  
get\_garch\_aic\_bic <- function(fit\_model) {  
 n <- length(fit\_model@fit$data)  
 loglik <- -0.5 \* (fit\_model@fit$sigma2 + log(2 \* pi) + fit\_model@fit$residuals^2 / fit\_model@fit$sigma2)  
 bic <- -2 \* sum(loglik) + log(n) \* (length(fit\_model@fit$coef) + 1)  
 aic <- -2 \* sum(loglik) + 2 \* (length(fit\_model@fit$coef) + 1)  
 return(c(AIC = aic, BIC = bic))  
}  
  
# Get AIC and BIC for each model  
sgarch\_aic\_bic <- get\_garch\_aic\_bic(sgarch\_model)  
sgarch\_sstd\_aic\_bic <- get\_garch\_aic\_bic(sgarch\_sstd\_model)  
gjrgarch\_aic\_bic <- get\_garch\_aic\_bic(gjrgarch\_model)  
tgarch\_aic\_bic <- get\_garch\_aic\_bic(tgarch\_model)  
  
# Print the AIC and BIC for each model  
print("sGARCH Model AIC and BIC:")

## [1] "sGARCH Model AIC and BIC:"

print(sgarch\_aic\_bic)

## AIC BIC   
## 14 -Inf

print("sGARCH-sstd Model AIC and BIC:")

## [1] "sGARCH-sstd Model AIC and BIC:"

print(sgarch\_sstd\_aic\_bic)

## AIC BIC   
## 12 -Inf

print("GJR-GARCH Model AIC and BIC:")

## [1] "GJR-GARCH Model AIC and BIC:"

print(gjrgarch\_aic\_bic)

## AIC BIC   
## 16 -Inf

print("TGARCH Model AIC and BIC:")

## [1] "TGARCH Model AIC and BIC:"

print(tgarch\_aic\_bic)

## AIC BIC   
## 14 -Inf