SAIGON HOTEL (VIETNAM)

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

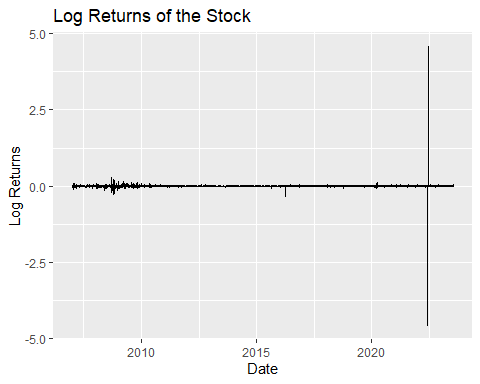
library(rugarch)

## 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/SAIGON HOTEL (Vietnam).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

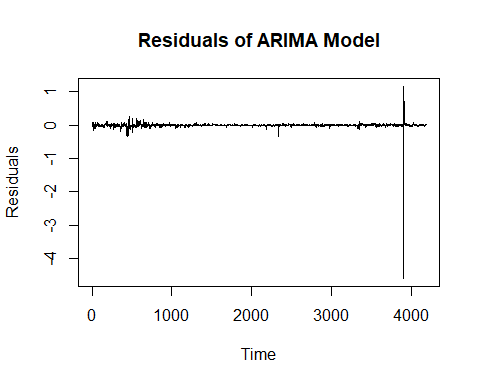
# Check if the time series is stationary based on the p-value  
if (p\_value < 0.05) {  
 print("The log returns time series is stationary.")  
} else {  
 print("The log returns time series is not stationary.")  
}

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

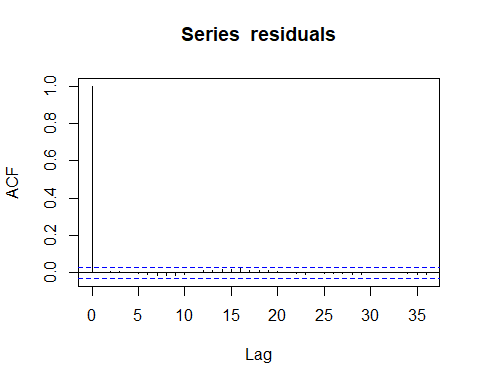
#The data is stationary, therefore, we continue modelling the arima  
arima\_model <- Arima(data$log\_returns, order = c(1, 0, 1))  
# Print the ARIMA model summary  
summary(arima\_model)

## Series: data$log\_returns   
## ARIMA(1,0,1) with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 mean  
## 0.0102 -0.7548 0e+00  
## s.e. 0.0208 0.0137 3e-04  
##   
## sigma^2 = 0.006725: log likelihood = 4529.25  
## AIC=-9050.49 AICc=-9050.48 BIC=-9025.14  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -5.548817e-06 0.0819788 0.0198669 NaN Inf 1.094385 -0.0001770683

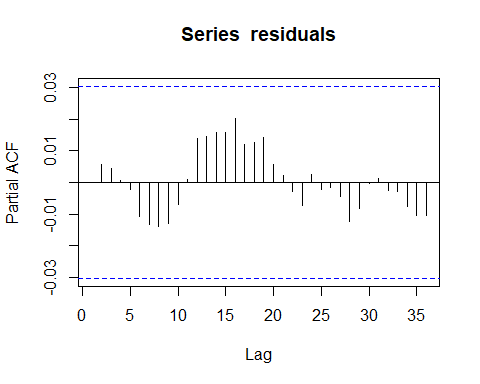
# Extract the residuals from the ARIMA model  
residuals <- residuals(arima\_model)  
# Plot the residuals to check for any patterns or trends  
plot(residuals, main = "Residuals of ARIMA Model", ylab = "Residuals")



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



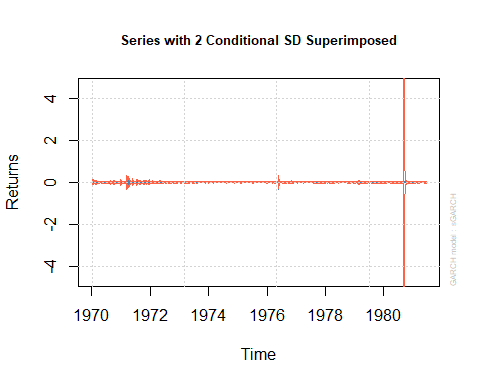
# Plot the PACF (Partial Autocorrelation Function) of the residuals  
pacf(residuals)



# 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)

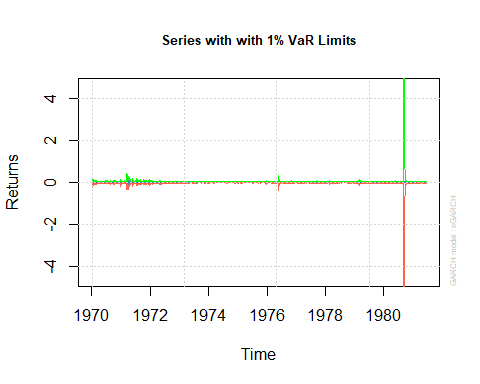
##   
## \*---------------------------------\*  
## \* GARCH Model Fit \*  
## \*---------------------------------\*  
##   
## Conditional Variance Dynamics   
## -----------------------------------  
## GARCH Model : sGARCH(1,1)  
## Mean Model : ARFIMA(1,0,1)  
## Distribution : norm   
##   
## Optimal Parameters  
## ------------------------------------  
## Estimate Std. Error t value Pr(>|t|)  
## mu 0.000248 0.000245 1.0120 0.31153  
## ar1 0.354144 0.134248 2.6380 0.00834  
## ma1 -0.229356 0.139841 -1.6401 0.10098  
## omega 0.000032 0.000002 14.6499 0.00000  
## alpha1 0.198831 0.017033 11.6732 0.00000  
## beta1 0.708976 0.016703 42.4459 0.00000  
##   
## Robust Standard Errors:  
## Estimate Std. Error t value Pr(>|t|)  
## mu 0.000248 0.000279 0.88932 0.373829  
## ar1 0.354144 0.153968 2.30012 0.021442  
## ma1 -0.229356 0.154773 -1.48189 0.138369  
## omega 0.000032 0.000022 1.47961 0.138978  
## alpha1 0.198831 0.060054 3.31087 0.000930  
## beta1 0.708976 0.059824 11.85107 0.000000  
##   
## LogLikelihood : 11598.14   
##   
## Information Criteria  
## ------------------------------------  
##   
## Akaike -5.5399  
## Bayes -5.5308  
## Shibata -5.5399  
## Hannan-Quinn -5.5366  
##   
## Weighted Ljung-Box Test on Standardized Residuals  
## ------------------------------------  
## statistic p-value  
## Lag[1] 0.02787 0.8674  
## Lag[2\*(p+q)+(p+q)-1][5] 0.04471 1.0000  
## Lag[4\*(p+q)+(p+q)-1][9] 0.05742 1.0000  
## d.o.f=2  
## H0 : No serial correlation  
##   
## Weighted Ljung-Box Test on Standardized Squared Residuals  
## ------------------------------------  
## statistic p-value  
## Lag[1] 0.0001255 0.9911  
## Lag[2\*(p+q)+(p+q)-1][5] 0.0006757 1.0000  
## Lag[4\*(p+q)+(p+q)-1][9] 0.0012252 1.0000  
## d.o.f=2  
##   
## Weighted ARCH LM Tests  
## ------------------------------------  
## Statistic Shape Scale P-Value  
## ARCH Lag[3] 0.0002763 0.500 2.000 0.9867  
## ARCH Lag[5] 0.0006561 1.440 1.667 1.0000  
## ARCH Lag[7] 0.0009775 2.315 1.543 1.0000  
##   
## Nyblom stability test  
## ------------------------------------  
## Joint Statistic: 7.3985  
## Individual Statistics:   
## mu 0.04464  
## ar1 1.45307  
## ma1 1.66785  
## omega 0.04996  
## alpha1 4.37350  
## beta1 0.16702  
##   
## 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 0.9406478 0.3469   
## Negative Sign Bias 0.0968487 0.9229   
## Positive Sign Bias 0.0002189 0.9998   
## Joint Effect 0.9212922 0.8203   
##   
##   
## Adjusted Pearson Goodness-of-Fit Test:  
## ------------------------------------  
## group statistic p-value(g-1)  
## 1 20 1832 0  
## 2 30 2020 0  
## 3 40 2219 0  
## 4 50 2409 0  
##   
##   
## Elapsed time : 1.744985

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

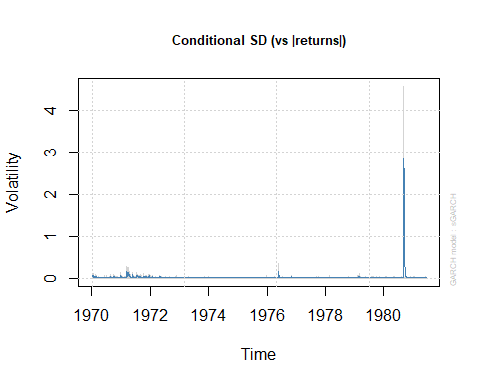


plot(garch\_model, which = 2)

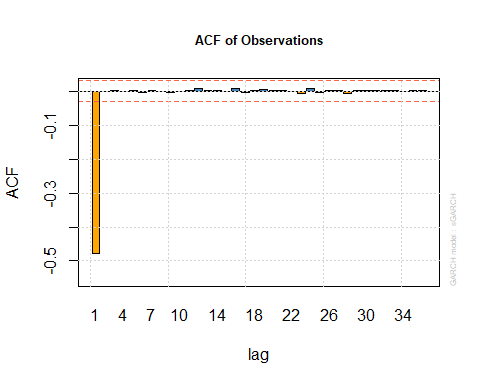
##   
## please wait...calculating quantiles...



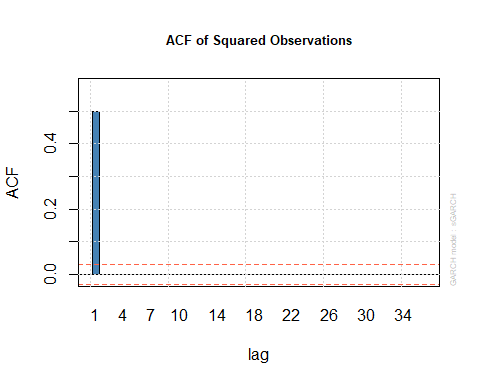
plot(garch\_model, which = 3)



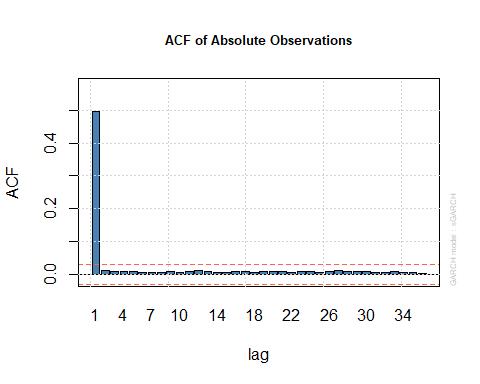
plot(garch\_model, which = 4)



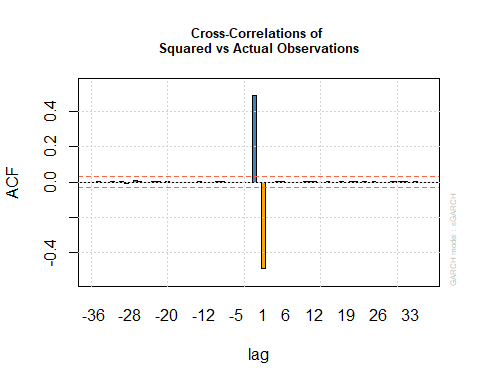
plot(garch\_model, which = 5)



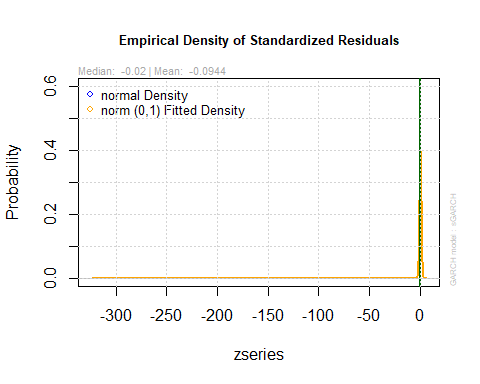
plot(garch\_model, which = 6)



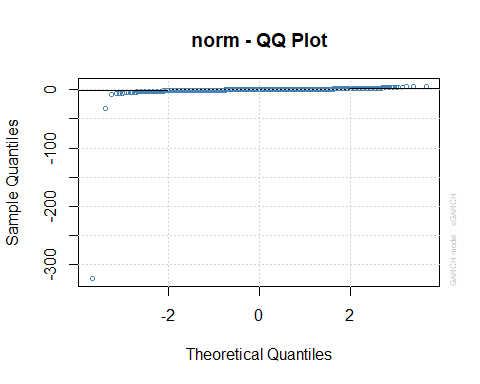
plot(garch\_model, which = 7)



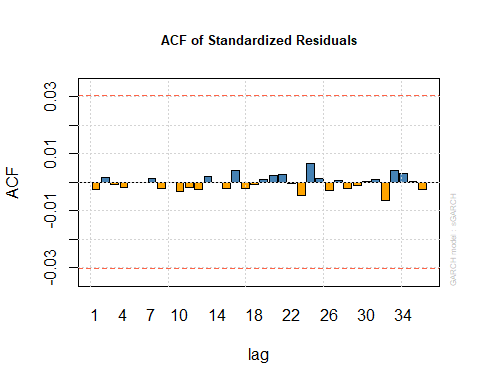
plot(garch\_model, which = 8)



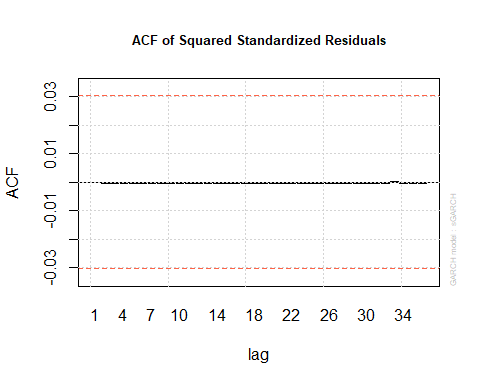
plot(garch\_model, which = 9)



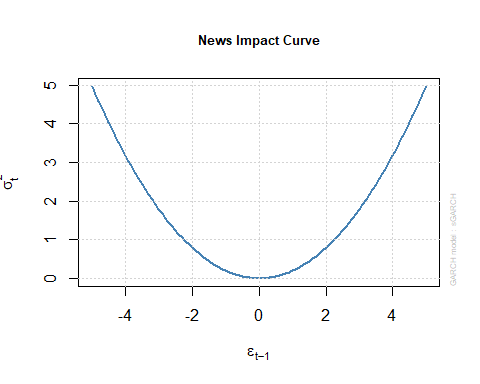
plot(garch\_model, which = 10)



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)

## Warning in .sgarchfit(spec = spec, data = data, out.sample = out.sample, :   
## ugarchfit-->warning: solver failer to converge.

# 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   
## 2 -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