Project econometrics

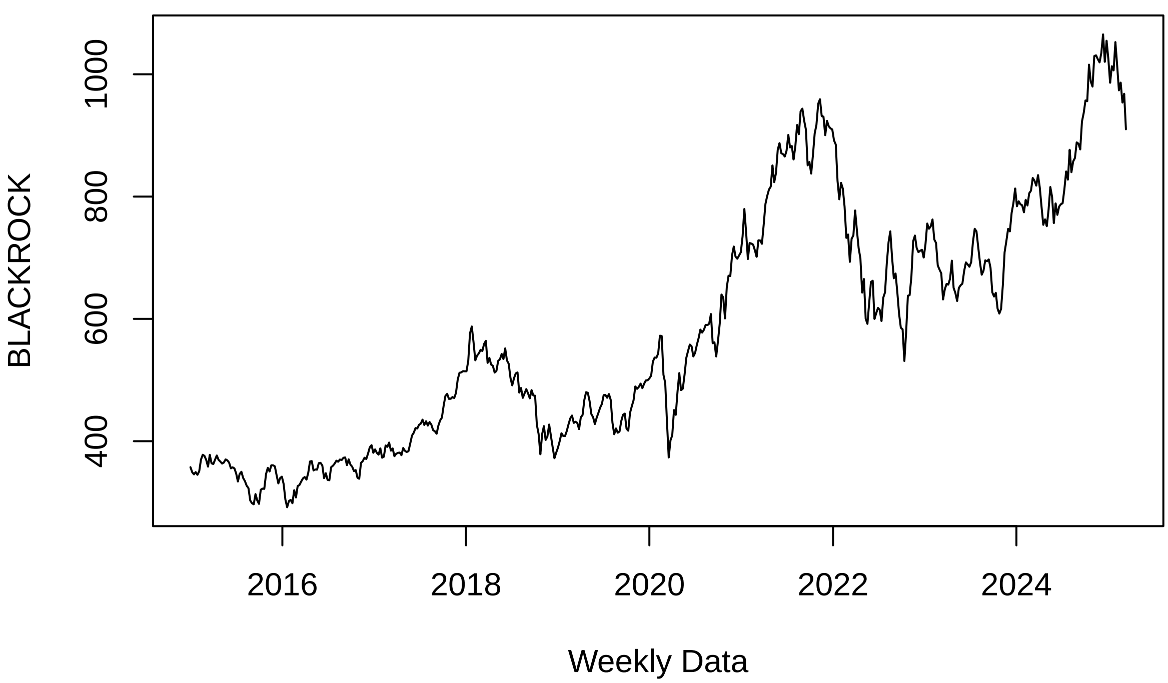
WEEK 1

Introduction

In this group project, we focus on developing and evaluating predictive models for the stock price of BLACKROCK. Our objective is to explore, on a weekly basis, progressively better forecasting approaches by testing and refining various modeling techniques. We utilize historical BLACKROCK price data spanning from December 31, 2014, to March 12, 2025, as the foundation for our analysis. By continuously assessing model performance, our goal is to identify and iterate towards more accurate and robust predictive tools over time. During our first week of work, we used a simple prediction model and examined descriptive statistics and basic plots. Firstly we loaded all the usefull packages and uploaded our data.

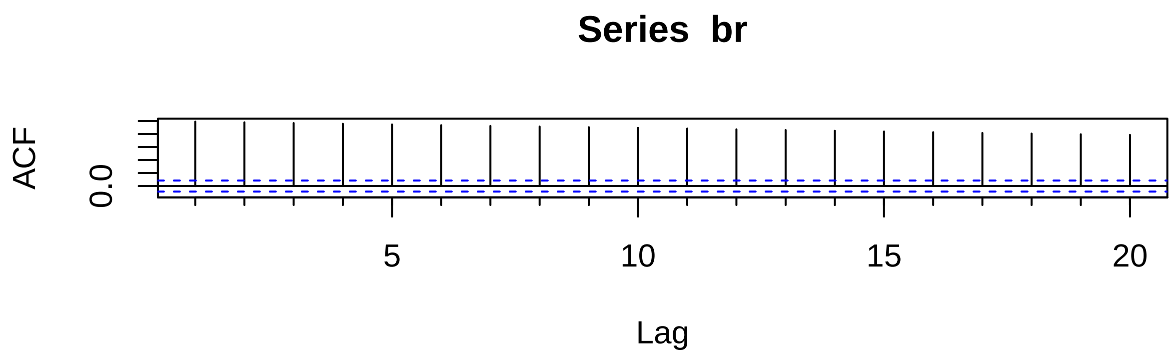
Visualization of Data

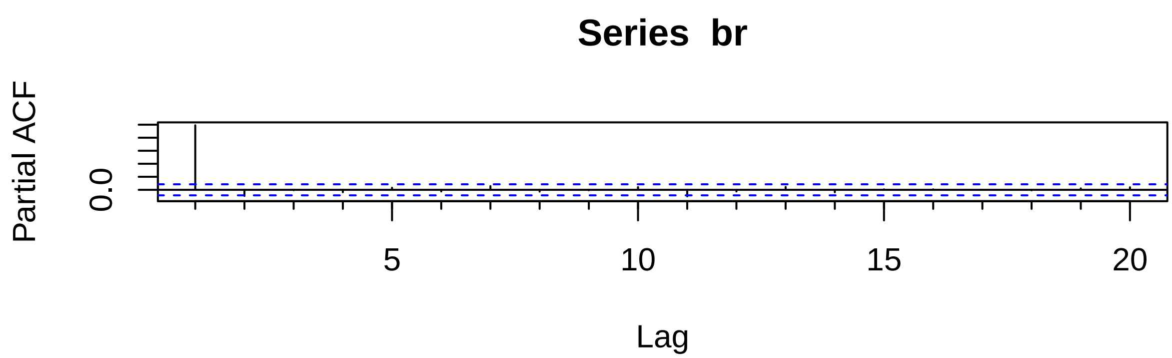
```
data = read_excel(here("blackrock.xlsx"), sheet = 1, col_names = TRUE)
```



After a simple visualization of our data, we observed the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to gather some initial insights about the appropriate model type and to evaluate whether any data transformations were needed.

```r
br = data$BLACKROCK
```





We can observe that the series is **non-stationary** due to the slow convergence of the ACF graph. After that, we checked for the presence of **non-stationarity** using mathematical approaches as well, and we reached the same conclusion.

```r
adfTest(br,lags=8,type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 8
##   STATISTIC:
##     Dickey-Fuller: -0.7995
##   P VALUE:
##     0.7619
##
## Description:
##  Tue May 20 20:31:54 2025 by user: User
```

```r
adfTest(br,lags=8,type="nc")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 8
##   STATISTIC:
##     Dickey-Fuller: 0.8364
##   P VALUE:
##     0.8829
##
## Description:
##  Tue May 20 20:31:54 2025 by user: User
```

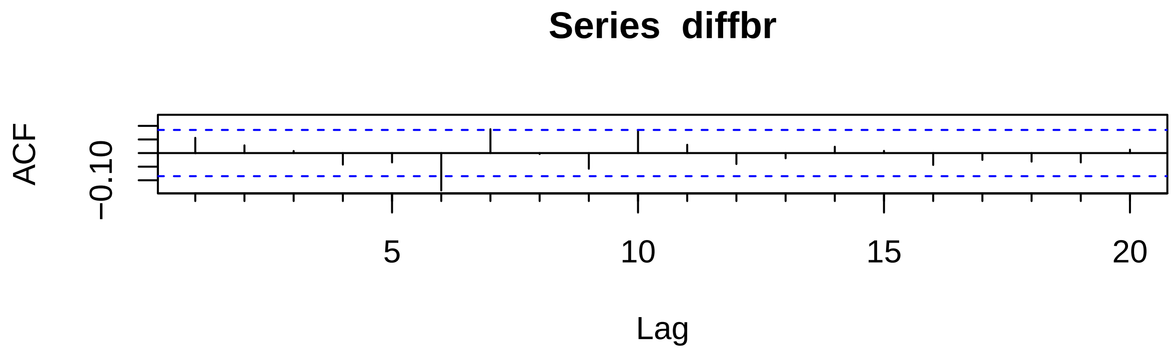
```r
adfTest(br,lags=8,type="ct")
```

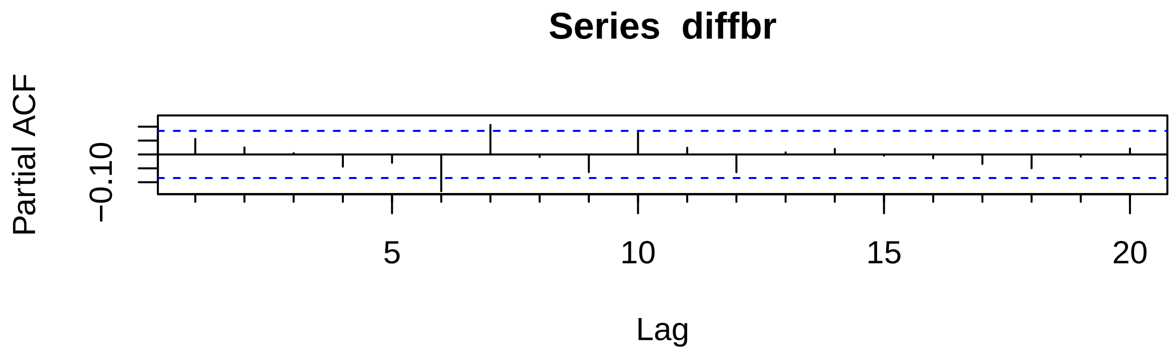
```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 8
##   STATISTIC:
##     Dickey-Fuller: -2.6243
##   P VALUE:
##     0.3141
##
## Description:
##  Tue May 20 20:31:54 2025 by user: User
```

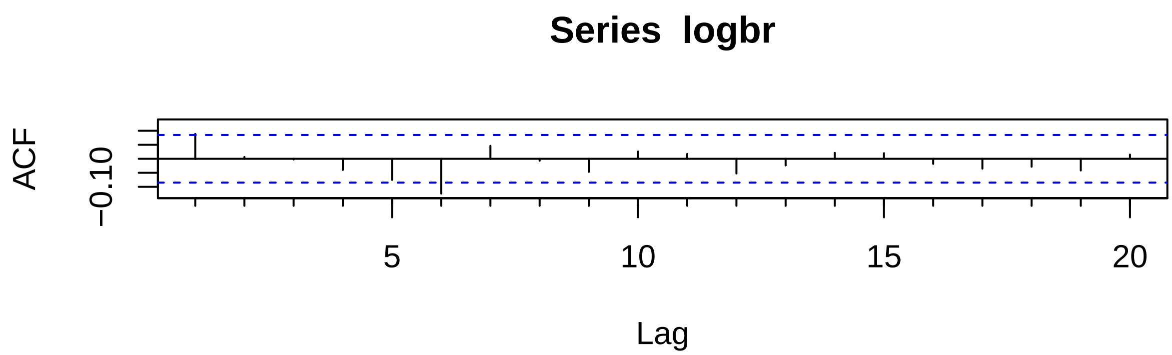
All accept H0 (presence of unit root) because p-value $> 5\%$, so we conclude that the series is not-stationary.

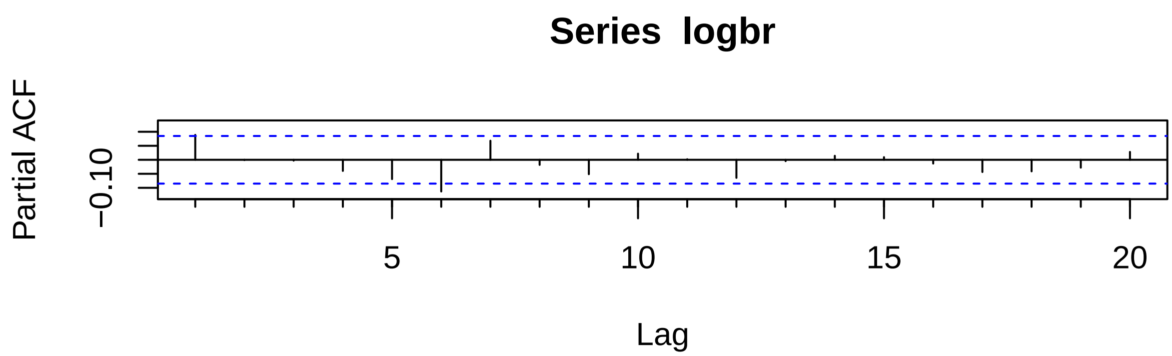
Transformation of the series

So, to obtain a stationary series, which implies the respect of consistent properties over time of data; we applied two transformation to the them and checked whether the results improved. The transformation are the first differenced series and first difference of logarithmic series.


Series diffbr
ACF
−0.10
5
10
15
20
Lag


Series diffbr
Partial ACF
−0.10
5
10
15
20
Lag





We can observe that the series appears to be stationary and quite similar between the two options. After reviewing some comparable statistics and tests like Box-Ljung or t-test, we can conclude that, for now, they can be used interchangeably. Therefore, we decided to proceed with the first differenced transformation.

Below, we present some descriptive statistics and basic tests.

```
basicStats(diffbr)
```

```
##                  diffbr
## nobs         532.000000
## NAs            0.000000
## Minimum      -64.879900
## Maximum       59.900100
## 1. Quartile   -8.507450
## 3. Quartile   11.107475
## Mean           1.038534
## Median         1.764900
## Sum          552.500000
## SE Mean        0.896543
## LCL Mean      -0.722672
## UCL Mean       2.799739
## Variance     427.615575
## Stdev         20.678868
## Skewness      -0.305413
## Kurtosis       0.951604
```

```r
Box.test(diffbr,lag=10,type='Ljung')
```

```
##
##  Box-Ljung test
##
## data:  diffbr
## X-squared = 23.652, df = 10, p-value = 0.008581
```

```r
t.test(diffbr)
```

```
##
##  One Sample t-test
##
## data:  diffbr
## t = 1.1584, df = 531, p-value = 0.2472
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.7226718  2.7997394
## sample estimates:
## mean of x
##  1.038534
```

The Box-Ljung test indicates that exist serial correlation among the residuals. The t-test on returns indicates that the mean is not significantly different from zero, suggesting that constant term may not be necessary in the ARMA model. As we can observe, we reject the null hypothesis of **no serial correlation**, p-value $= 0.008 < 5\%$, tested by the Box-Ljung test. Unfortantly, we do not reject the null hypothesis of the **mean being equal to 0**, p-value $= 0.2472 > 5\%$, tested by the t-test.

```r
m1=ar(diffbr, aic=TRUE, order.max=10)
m1$order
```

```
## [1] 7
```

Finally, the AR test that uses the AIC value to choose the best model suggests using an AR model of order 7.

Hypothesized models

The first model we have hypothesized was the ARIMA(7,1,0) suggested by the order of ar function.

```r
mod1 = arima(br,order=c(7,1,0),include.mean=F)
summary(mod1)
```

```
##
## Call:
## arima(x = br, order = c(7, 1, 0), include.mean = F)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5      ar6     ar7
##       0.0656  0.0258  0.0168  -0.0395  -0.0208  -0.1398  0.1146
```

```
## s.e.  0.0433  0.0431  0.0431   0.0431   0.0432   0.0434  0.0438
##
## sigma^2 estimated as 412.3:  log likelihood = -2356.75,  aic = 4729.5
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.027483 20.28531 15.08175 0.1164675 2.604512 0.9939781
##                      ACF1
## Training set -0.001795314
```

```
confint(mod1)
```

```
##           2.5 %      97.5 %
## ar1 -0.01936784  0.15047486
## ar2 -0.05861520  0.11025037
## ar3 -0.06776860  0.10135139
## ar4 -0.12389235  0.04496040
## ar5 -0.10556623  0.06392235
## ar6 -0.22477331 -0.05476216
## ar7  0.02875549  0.20039630
```

We have now considered lags up to 16, meaning we are investigating whether past values from up to 4 months ago may affect the present data..

```
jarqueberaTest(mod1$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 19.0105
##   P VALUE:
##     Asymptotic p Value: 7.446e-05
```

```
Box.test(mod1$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod1$residuals
## X-squared = 7.9894, df = 16, p-value = 0.9492
```

```
archtest(as.vector(mod1$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod1$residuals)
## statistic = 70.225, lag = 16, p-value = 9.112e-09
## alternative hypothesis: ARCH effects of order 16 are present
```

The hypothesis of normally distributed residuals is rejected, p-value = 0,00007 < 5%, ARCH effects are present, p-value = $9.11e^{-09}$ < 5% so we reject the null hypothesis of presence of ARCH effect, and residuals appear to be non-autocorrelated, p-value = 0,9492 > 5%.

Thanks to the confint function, we observed that only two parameters ***the sixth and seventh lags*** are statistically significant. Therefore, we modified our model to include only these two parameters.

```
c1 = c(0,0,0,0,0,NA,NA)
mod1m = arima(br,order=c(7,1,0),include.mean=F, fixed = c1)
summary(mod1m)
```

```
##
## Call:
## arima(x = br, order = c(7, 1, 0), include.mean = F, fixed = c1)
##
## Coefficients:
##       ar1  ar2  ar3  ar4  ar5      ar6     ar7
##         0    0    0    0    0  -0.1448  0.1038
## s.e.    0    0    0    0    0   0.0434  0.0435
##
## sigma^2 estimated as 415.4:  log likelihood = -2358.76,  aic = 4723.53
##
## Training set error measures:
##                   ME     RMSE      MAE      MPE     MAPE      MASE       ACF1
## Training set 1.09966 20.36254 15.10379 0.123671 2.612463 0.9954305 0.06504822
```

```
confint(mod1m)
```

```
##           2.5 %      97.5 %
## ar1          NA          NA
## ar2          NA          NA
## ar3          NA          NA
## ar4          NA          NA
## ar5          NA          NA
## ar6 -0.22977272 -0.05976766
## ar7  0.01841098  0.18910852
```

```
jarqueberaTest(mod1m$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 20.3169
##   P VALUE:
##     Asymptotic p Value: 3.875e-05
```

```
Box.test(mod1m$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod1m$residuals
## X-squared = 10.866, df = 16, p-value = 0.8177
```

```r
archtest(as.vector(mod1m$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod1m$residuals)
## statistic = 69.603, lag = 16, p-value = 1.171e-08
## alternative hypothesis: ARCH effects of order 16 are present
```

This modified model does not rely on different assumptions, but it shows slightly better error metrics.

Finally, we observed and compared the model suggested by the **auto.arima** function applied to the differenced transformation of the data. The auto.arima() function automatically finds the best ARIMA model for a given time series, returning the model with the lowest AICc, which balances goodness of fit and model complexity.

```r
auto.arima(diffbr)
```

```
## Series: diffbr
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 427.9:  log likelihood = -2366.53
## AIC=4735.07   AICc=4735.08   BIC=4739.34
```

```r
mod2= arima(br, order=c(0,1,0), include.mean = F)
summary(mod2)
```

```
##
## Call:
## arima(x = br, order = c(0, 1, 0), include.mean = F)
##
##
## sigma^2 estimated as 427.9:  log likelihood = -2366.53,  aic = 4735.07
##
## Training set error measures:
##                    ME    RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 1.037256 20.6661 15.14532 0.1155377 2.606986 0.998168 0.0560253
```

```r
jarqueberaTest(mod2$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
```

```
##   STATISTIC:
##     X-squared: 29.4012
##   P VALUE:
##     Asymptotic p Value: 4.127e-07
```

```r
Box.test(mod2$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod2$residuals
## X-squared = 26.687, df = 16, p-value = 0.04509
```

```r
archtest(as.vector(mod2$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod2$residuals)
## statistic = 73.256, lag = 16, p-value = 2.666e-09
## alternative hypothesis: ARCH effects of order 16 are present
```

This time, we rejected all the hypotheses at a 5% level of significance, including the null hypothesis of non-autocorrelated residuals.

At this stage, model performance can be evaluated by comparing AIC and BIC scores.

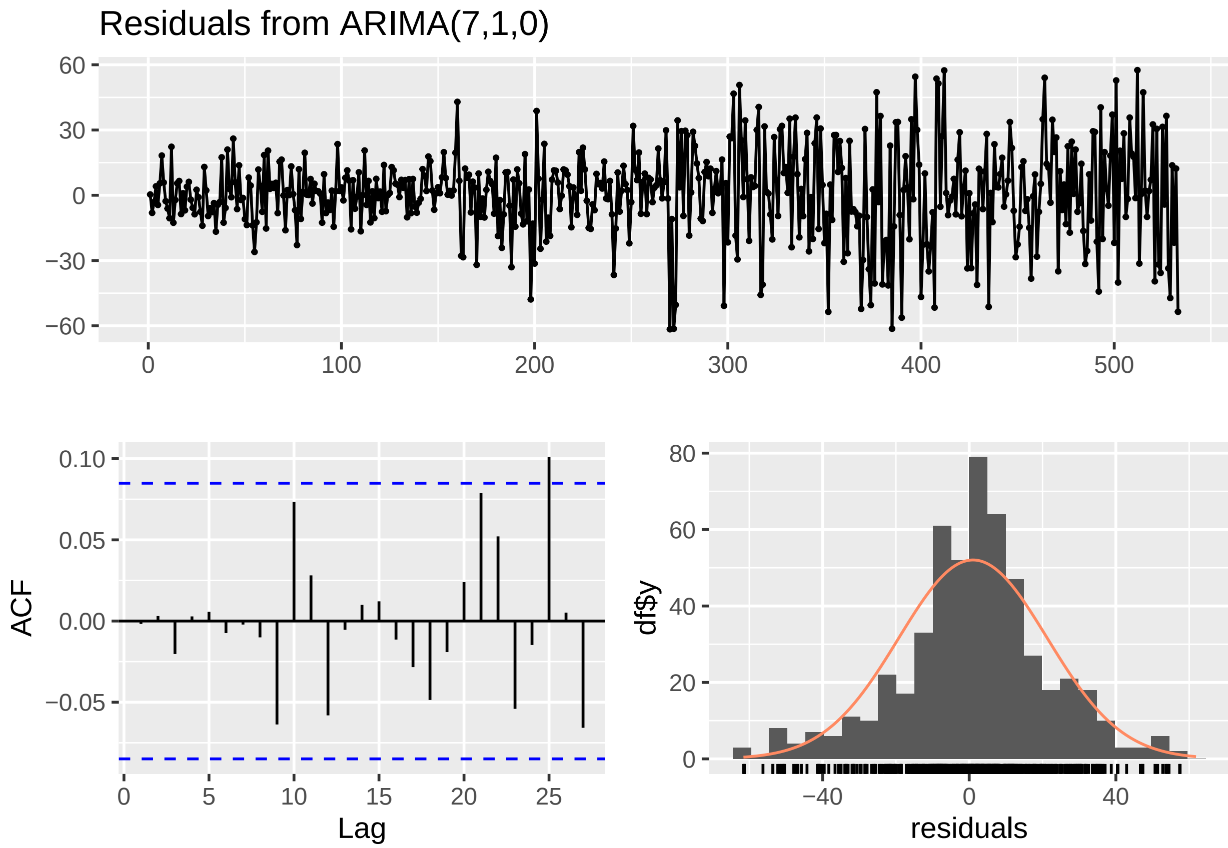
```r
cbind(AIC(mod1), AIC(mod1m), AIC(mod2))
```

```
##          [,1]     [,2]     [,3]
## [1,] 4729.497 4723.529 4735.068
```

```r
cbind(BIC(mod1), BIC(mod1m), BIC(mod2))
```

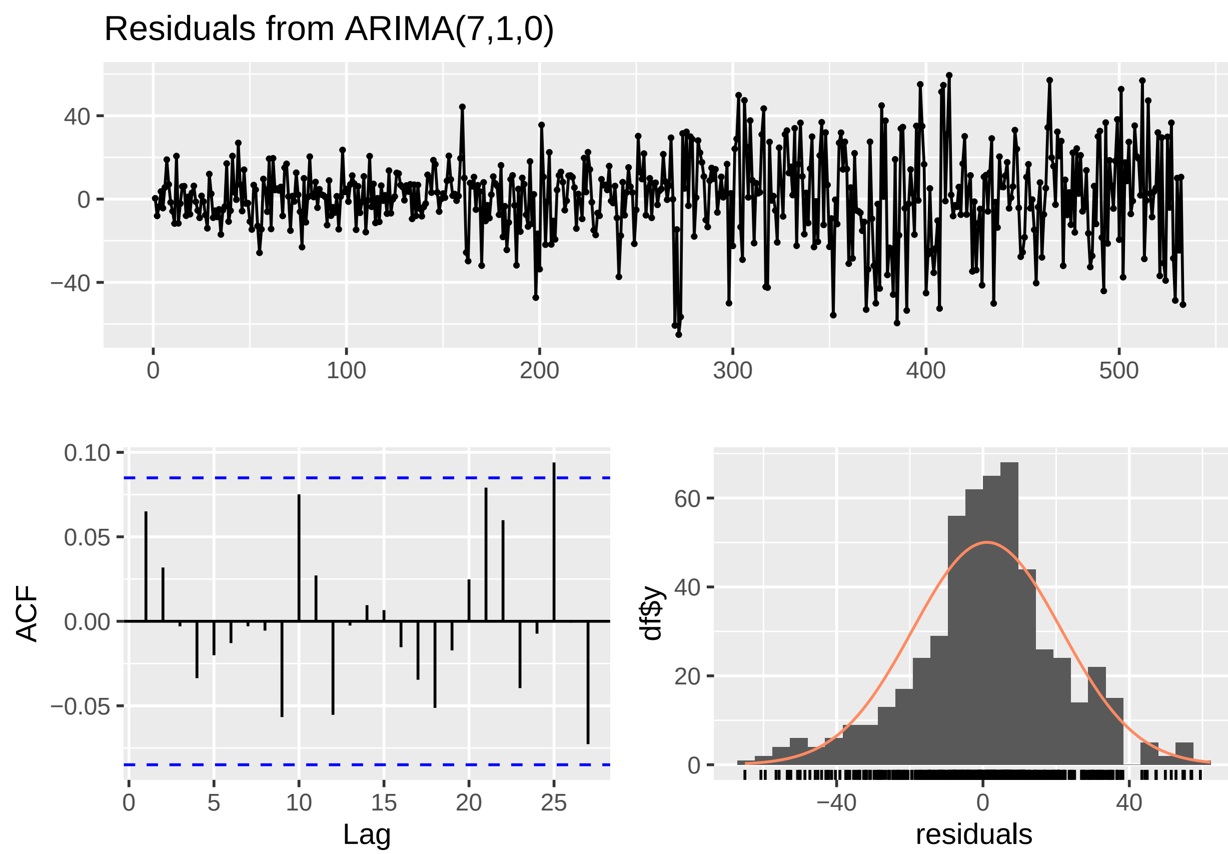
```
##         [,1]     [,2]     [,3]
## [1,] 4763.71 4736.359 4739.344
```

```r
checkresiduals(mod1)
```



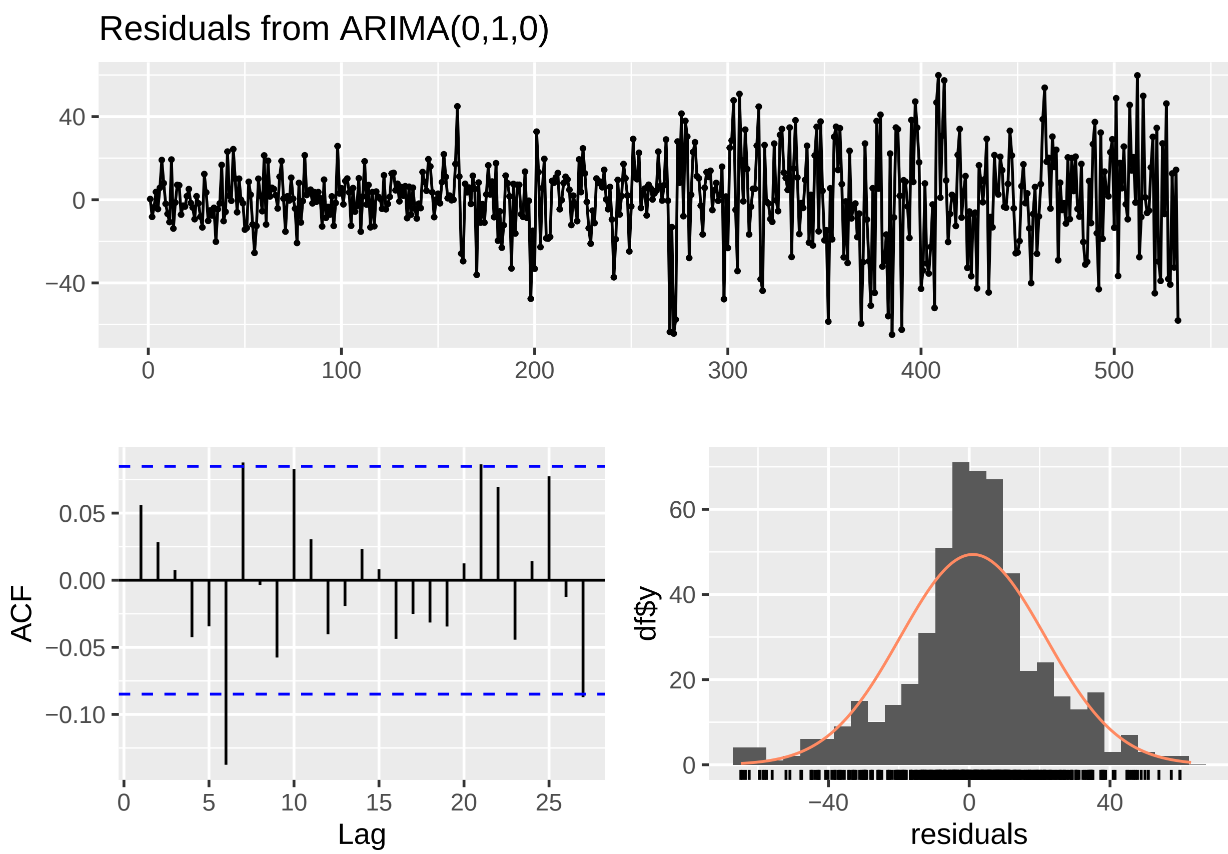

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(7,1,0)
## Q* = 5.4862, df = 3, p-value = 0.1395
##
## Model df: 7.   Total lags used: 10
```

```
checkresiduals(mod1m)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(7,1,0)
## Q* = 8.5808, df = 3, p-value = 0.03542
##
## Model df: 7.   Total lags used: 10
```

```
checkresiduals(mod2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 23.705, df = 10, p-value = 0.008424
##
## Model df: 0.   Total lags used: 10
```

All the scores suggest us that the best predictive model among those is the ARIMA(7,1,0) modified which consider only 2 parameters statistically significant.

Forecast

Finally, using the model specified above, we are able to forecast the price of **BLACKROCK** for 19/03/2025.

```
f = forecast(mod1m, h = 1)
f
```

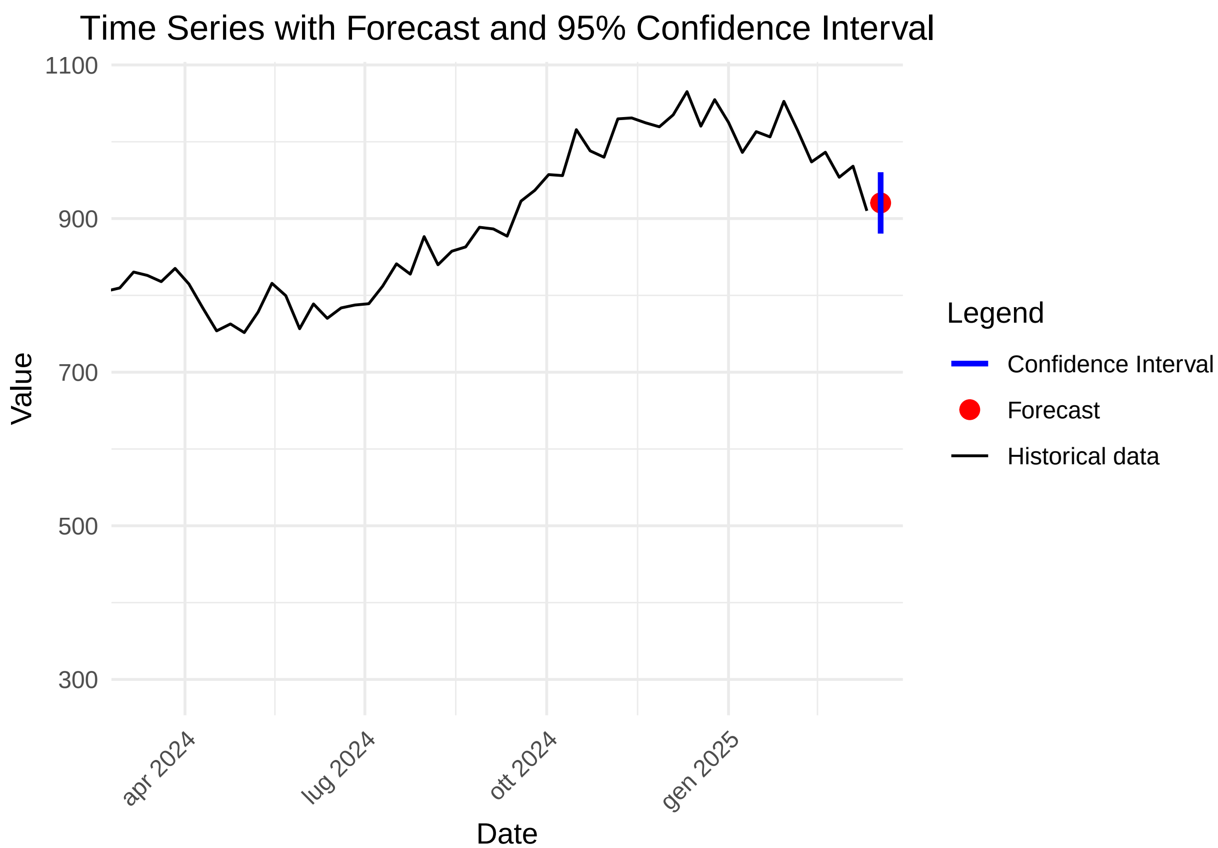
```
##     Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## 534       920.3783 894.2581 946.4984 880.4309 960.3256
```

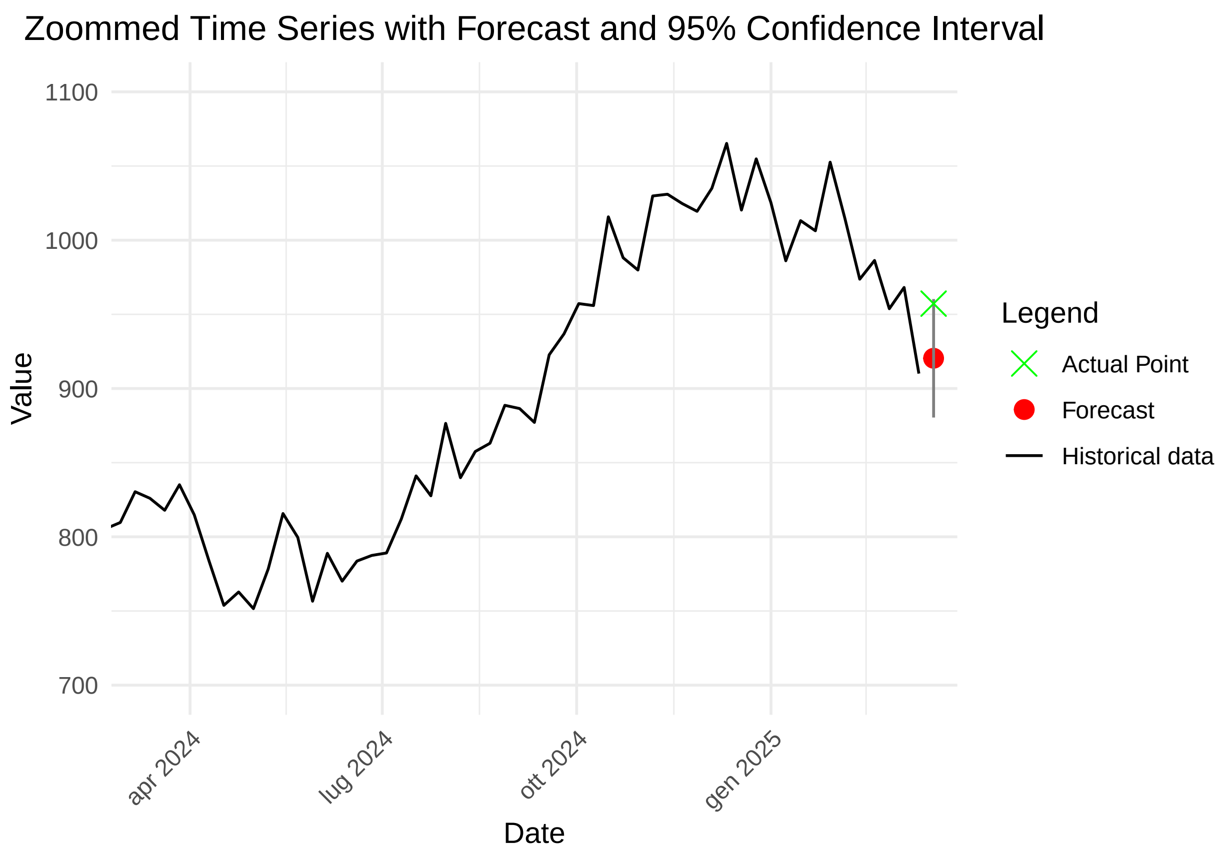
```
predicted_price <- 920.3783
real_price <- 957.23
```

```r
abs_error <- abs(real_price - predicted_price)
perc_error <- (abs_error / real_price) * 100
```

```
## Absolute error first week prediction: 36.8517
```

```
## Percentage error first week prediction: 3.849827
```





WEEK 2

During the second week, our knowledge of more complex models was not sufficient, so we decided to compare our model from the previous week with a differently specified model, using a train-test split on the dataset. So using the same packages of last week we uploaded our dataset and transformed values in logarithmic return divided in training and test (90%-10%).

```
data <- read_excel(here("blackrock.xlsx"), col_names = TRUE, sheet =2)

data$log_price <- log(data$BLACKROCK)
data$log_return <- c(NA, diff(data$log_price))
data_clean <- data[-1, ]

train_size <- floor(0.9 * nrow(data_clean))
train <- data_clean[1:train_size, ]
test <- data_clean[(train_size + 1):nrow(data_clean), ]

br <- data$BLACKROCK
brtr <- train$BLACKROCK
brte <- test$BLACKROCK
```

We decided to use a function that can found the ebst model, with the lowest AIC value among a specified rango of parameter, range obtained observing the ACF and PACF of series.

```r
select_best_arima <- function(series, max_p = 7, max_q = 7) {
  best_aic <- Inf
  best_model <- NULL
  best_order <- c(0, 1, 0)

  for (p in 0:max_p) {
    for (q in 0:max_q) {
      result <- tryCatch({
        model <- arima(series, order = c(p, 1, q))
        aic <- AIC(model)
        if (aic < best_aic) {
          best_aic <- aic
          best_model <- model
          best_order <- c(p, 1, q)
        }
      })
    }
  }

  if (!is.null(best_model)) {
    cat("Best ARIMA model found: (", paste(best_order, collapse = ","), ")\n")
    cat("AIC:", best_aic, "\n")
    return(best_model)
  } else {
    cat("No valid model found.\n")
    return(NULL)
  }
}

best_model <- select_best_arima(brtr, max_p = 7, max_q = 7)
```

```
## Warning in arima(series, order = c(p, 1, q)): possibile errore di convergenza:
## optim ha restituito codice = 1
## Warning in arima(series, order = c(p, 1, q)): possibile errore di convergenza:
## optim ha restituito codice = 1
## Warning in arima(series, order = c(p, 1, q)): possibile errore di convergenza:
## optim ha restituito codice = 1
## Warning in arima(series, order = c(p, 1, q)): possibile errore di convergenza:
## optim ha restituito codice = 1
## Warning in arima(series, order = c(p, 1, q)): possibile errore di convergenza:
## optim ha restituito codice = 1
```

```
## Best ARIMA model found: ( 7,1,7 )
## AIC: 4196.875
```

Thanks to this function we obtained as best model ARIMA(7,1,7), with the best AIC value but with a problem of converge and low parsimony due to a large number of parameter. Anywhere, now we check all the test of Normlaity, serial-correlation and arch effect on that series.

```r
mod2 <- arima(brtr, order=c(7,1,7),include.mean=F)
```

```
## Warning in arima(brtr, order = c(7, 1, 7), include.mean = F): possibile errore
## di convergenza: optim ha restituito codice = 1
```

```r
jarqueberaTest(mod2$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 41.6588
##   P VALUE:
##     Asymptotic p Value: 8.993e-10
```

```r
Box.test(mod2$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod2$residuals
## X-squared = 9.3473, df = 16, p-value = 0.8984
```

```r
archtest(as.vector(mod2$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod2$residuals)
## statistic = 68.199, lag = 16, p-value = 2.059e-08
## alternative hypothesis: ARCH effects of order 16 are present
```

The hypothesis of normally distributed residuals is rejected, p-value $= 8.993e^{-10} < 5\%$, ARCH effects of order 16 are present, p-value $= 2.059e^{-08} < 5\%$ so we reject the null hypothesis of presence of ARCH effect, and residuals appear to be non-autocorrelated, p-value $=$ 0,8984 $> 5\%$.

```r
confint(mod2)
```

```
##          2.5 %      97.5 %
## ar1 -0.6341128 -0.04201856
## ar2 -1.0779846 -0.56645883
## ar3 -0.2548224  0.53216144
## ar4 -0.5943669  0.14492426
## ar5 -0.9111353 -0.21325891
## ar6 -0.7369029 -0.46922997
## ar7 -0.9057380 -0.35091462
## ma1  0.1819671  0.73706749
## ma2  0.7073657  1.07764067
## ma3 -0.3716359  0.31310927
## ma4 -0.0515369  0.58385741
## ma5  0.2915330  0.96872926
## ma6  0.5289555  0.75698578
## ma7  0.4894319  1.02457885
```

We can observe thanks to the confint function, that some coefficient aren't significative, so as for the mod1 of first week we take them equal to 0.

```r
c2 = c(NA, NA, 0, 0, NA, NA, NA, NA, NA, 0, 0, NA, NA, NA)
mod2m = arima(brtr, order=c(7,1,7),include.mean=F, fixed = c2)
```

```
## Warning in arima(brtr, order = c(7, 1, 7), include.mean = F, fixed = c2):
## alcuni parametri AR sono stati fissati: imposto transform.pars = FALSE
```

```r
confint(mod2m)
```

```
##           2.5 %      97.5 %
## ar1  0.02400115  0.32714449
## ar2 -0.66867764 -0.42134745
## ar3          NA          NA
## ar4          NA          NA
## ar5 -0.68408172 -0.21510381
## ar6 -0.22767907  0.16788564
## ar7 -0.98689911 -0.49165203
## ma1 -0.21355867  0.05496919
## ma2  0.43357412  0.64789528
## ma3          NA          NA
## ma4          NA          NA
## ma5  0.14953813  0.59528680
## ma6 -0.17616418  0.15603128
## ma7  0.65034147  1.02910464
```

I remove one more time the coefficient no more significative.

```r
c3 = c(NA, NA, 0, 0, NA, 0, NA, 0, NA, 0, 0, NA, 0, NA)
mod2m = arima(brtr, order=c(7,1,7),include.mean=F, fixed = c3)
```

```
## Warning in arima(brtr, order = c(7, 1, 7), include.mean = F, fixed = c3):
## alcuni parametri AR sono stati fissati: imposto transform.pars = FALSE

## Warning in log(s2): Si è prodotto un NaN
```

```r
confint(mod2m)
```

```
##           2.5 %     97.5 %
## ar1  0.01243985  0.1077540
## ar2 -0.67366649 -0.3323098
## ar3          NA         NA
## ar4          NA         NA
## ar5 -0.68870015 -0.5831660
## ar6          NA         NA
## ar7 -1.01420717 -0.6098892
## ma1          NA         NA
## ma2  0.31636128  0.6303469
## ma3          NA         NA
## ma4          NA         NA
## ma5  0.54403484  0.6746226
## ma6          NA         NA
## ma7  0.72760145  1.0497194
```

```r
jarqueberaTest(mod2m$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 45.1358
##   P VALUE:
##     Asymptotic p Value: 1.581e-10
```

```r
Box.test(mod2m$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod2m$residuals
## X-squared = 11.294, df = 16, p-value = 0.791
```

```r
archtest(as.vector(mod2m$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod2m$residuals)
## statistic = 67.544, lag = 16, p-value = 2.675e-08
## alternative hypothesis: ARCH effects of order 16 are present
```

The hypothesis of normally distributed residuals is rejected, p-value $= 1.581e^{-10} < 5\%$, ARCH effects of order 16 are present, p-value $= 2.675e^{-08} < 5\%$ so we reject the null hypothesis of presence of ARCH effect, and residuals appear to be non-autocorrelated, p-value $=$ 0,791 $> 5\%$.

Last week Model

As a comparative model, we used the one obtained during the previous week: an ARIMA(7,1,0) modified to include only the two statistically significant parameters. Even with the addition of a new observation, it maintains the same assumptions

```r
c1 = c(0,0,0,0,0,NA,NA)
mod1m = arima(br,order=c(7,1,0),include.mean=F, fixed = c1)
```

```
## Warning in arima(br, order = c(7, 1, 0), include.mean = F, fixed = c1): alcuni
## parametri AR sono stati fissati: imposto transform.pars = FALSE
```

```r
summary(mod1m)
```

```
##
## Call:
## arima(x = br, order = c(7, 1, 0), include.mean = F, fixed = c1)
##
## Coefficients:
##       ar1  ar2  ar3  ar4  ar5      ar6     ar7
##         0    0    0    0    0  -0.1515  0.1118
## s.e.    0    0    0    0    0   0.0433  0.0434
##
## sigma^2 estimated as 417.1:  log likelihood = -2364.3,  aic = 4734.6
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE    MAPE     MASE       ACF1
## Training set 1.164417 20.40432 15.1572 0.1304314 2.61771 0.995014 0.05623434
```

```r
confint(mod1m)
```

```
##           2.5 %      97.5 %
## ar1          NA          NA
## ar2          NA          NA
## ar3          NA          NA
## ar4          NA          NA
## ar5          NA          NA
## ar6 -0.23638520 -0.06669747
## ar7  0.02678415  0.19689312
```

At this stage, model performance can be evaluated by comparing RSME thanks to the division of dataset in training and test part.

```r
forecast1 <- predict(mod1m, n.ahead = length(brte))$pred
forecast2 <- predict(mod2, n.ahead = length(brte))$pred
forecast3 <- predict(mod2m, n.ahead = length(brte))$pred
```

```
## Warning in predict.Arima(mod2m, n.ahead = length(brte)): La parte MA del
## modello non è invertibile
```

```r
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

rmse1 <- rmse(brte, forecast1)
rmse2 <- rmse(brte, forecast2)
rmse3 <- rmse(brte, forecast3)

cat("RMSE Modello 1:", rmse1, "\n")
```

```
## RMSE Modello 1: 111.5859
```

```r
cat("RMSE Modello 2:", rmse2, "\n")
```

```
## RMSE Modello 2: 125.7274
```

```r
cat("RMSE Modello 3:", rmse3, "\n")
```

```
## RMSE Modello 3: 126.0999
```

Finally, we choose to use as for previous week the model ARIMA(7,1,0) modified to include only the two statistically significant parameters, because even it has higher AIC(Akaike Information Criteria) value, it has a lower value of RSME(root squared mean error), and we prefer to use a parsimonius model instead a model that may has problem of convergence and higher RMSE.

Forecast

At last, using the model specified above, we are able to forecast the price of **BLACKROCK** for 26/03/2025.

```r
f = forecast(mod1m, h = 1)
f
```

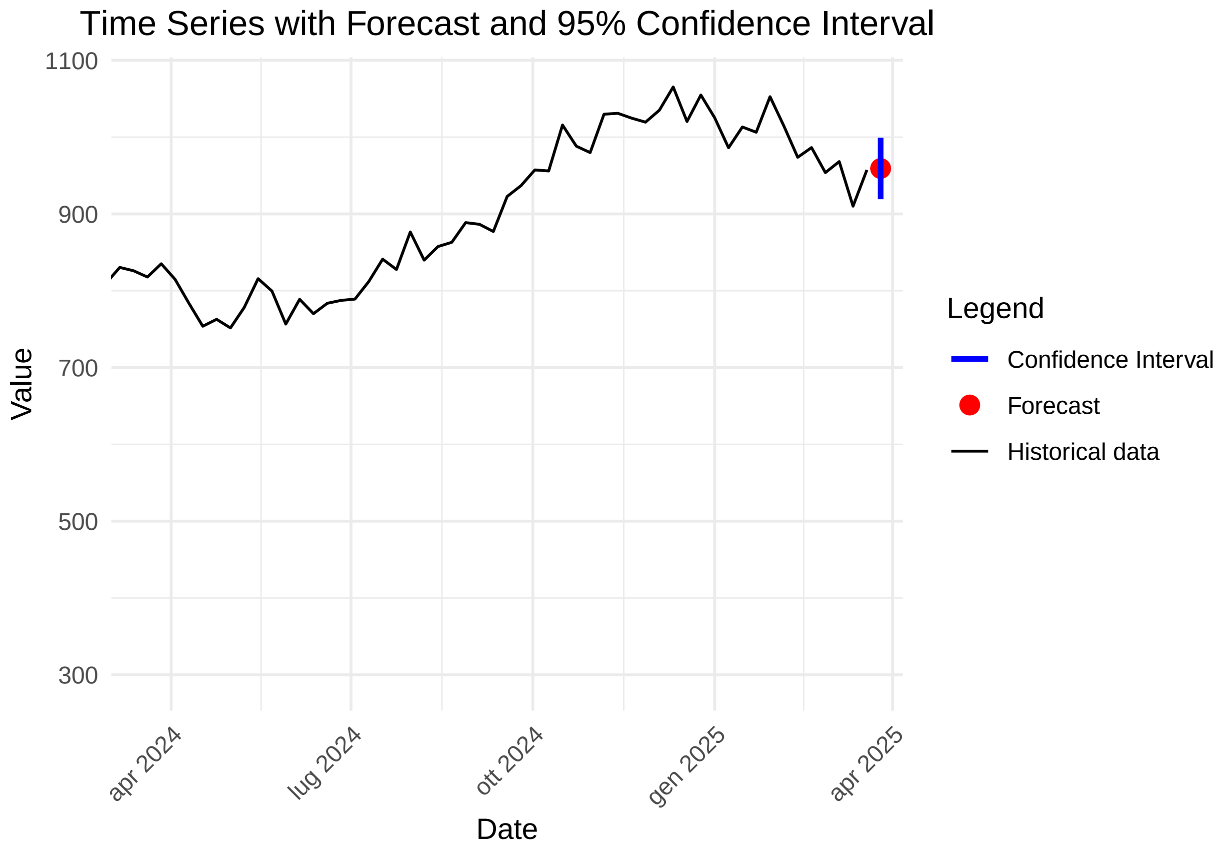
```
##     Point Forecast    Lo 80    Hi 80  Lo 95    Hi 95
## 535       959.1492 932.9755 985.3229 919.12 999.1784
```

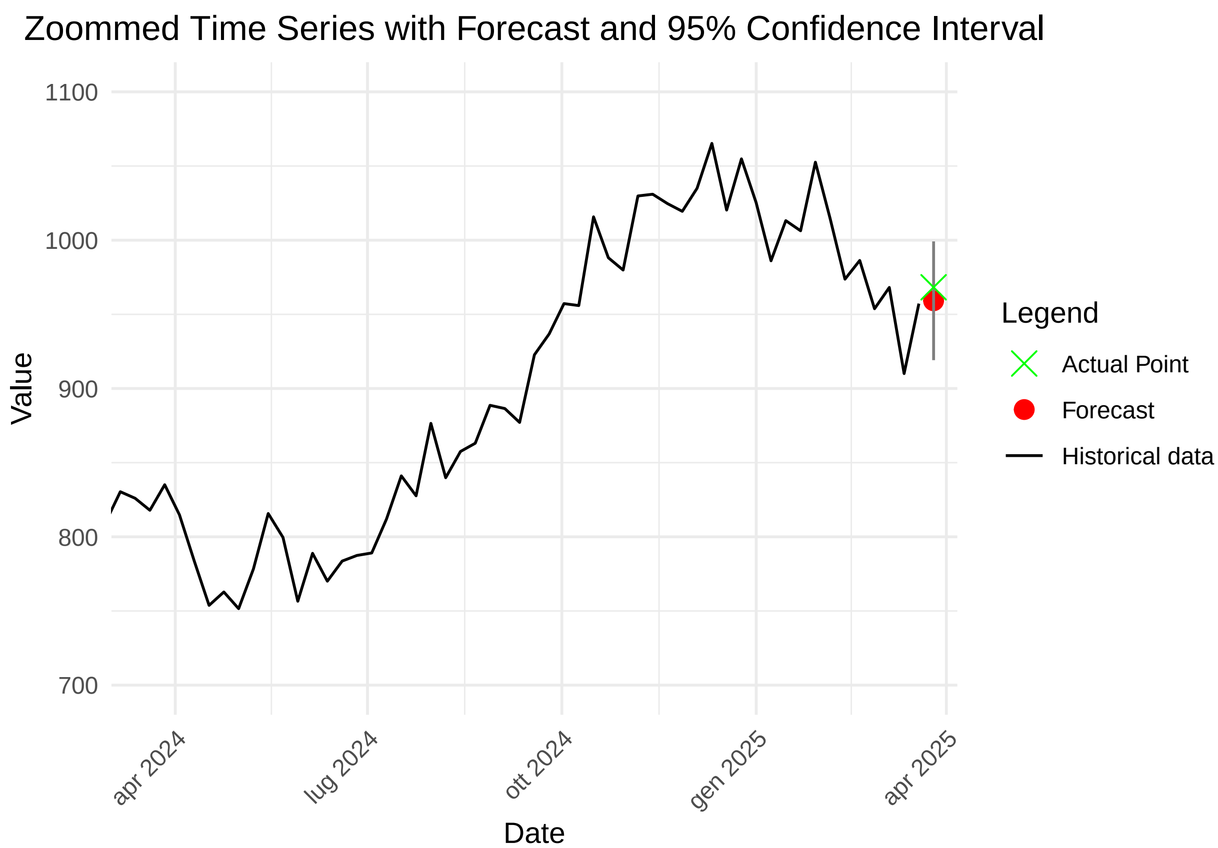
```r
predicted_price <- 959.15
real_price <- 968.24
```

```r
abs_error <- abs(real_price - predicted_price)
perc_error <- (abs_error / real_price) * 100
```

```
## Absolute error second week prediction: 9.09
```

```
## Percentage error second week prediction: 0.9388168
```


Time Series with Forecast and 95% Confidence Interval
1100
900
700
500
300
Value
apr 2024
lug 2024
ott 2024
gen 2025
apr 2025
Date
Legend
Confidence Interval
Forecast
Historical data



WEEK 3

Introduction

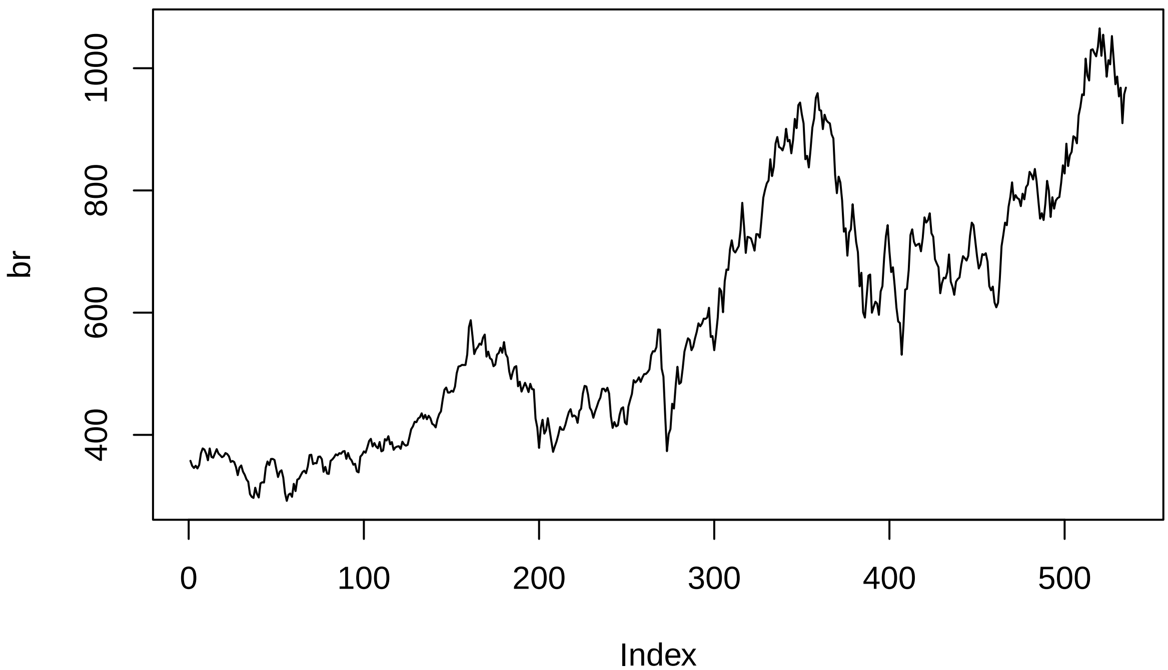
In the third week we decided to build upon the foundational ARIMA models developed in Weeks 1 and 2 by incorporating volatility modeling through a GARCH component. ARIMA models in fact, can capture the linear dependencies in the time series, but they do not account for time-varying volatility which is a common characteristic in financial data.

```
data = read_excel(here("blackrock.xlsx"), sheet = 3, col_names = TRUE)[c(1,2)]
```

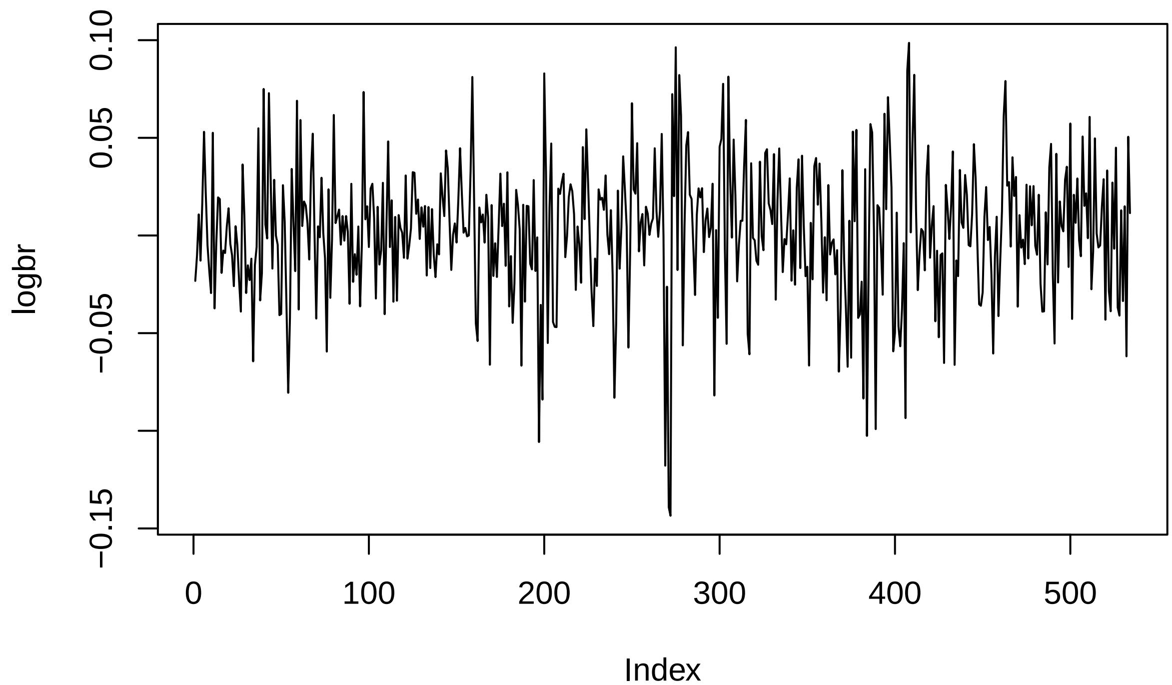
Data Visualization and Transformation

Initially, we plotted the raw price series to examine the trends over time, observing a volatility in the short term with more pronounced fluctuations, but an overall stable and upward trend in the long term. Subsequently, we calculated the logarithmic returns, a key transformation to adjust the data for more accurate and relevant analysis. This transformation allows us to focus on relative changes over time, making the model more suitable for time series forecasting and volatility modeling, both crucial for understanding and anticipating market movements.

```
br = data$BLACKROCK
plot(br, type='l')
```




```
logbr = diff(log(br))
plot(logbr, type='l')
```



ARIMA Estimation

t-test

In this part of the analysis, we perform a t-test on the logarithmic returns series (logbr) to check if the mean is significantly different from zero. If the mean is zero, it implies that there is no significant constant component in the series, which would be ideal for a trend-free predictive model.

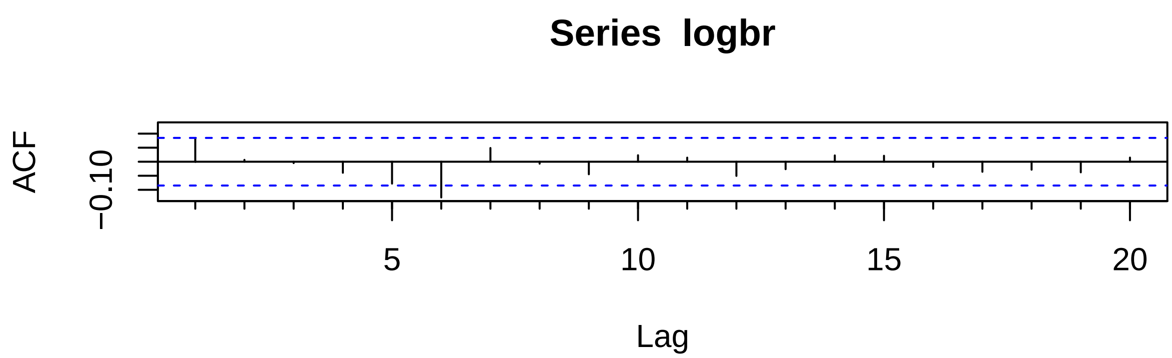
```
t.test(logbr)
```

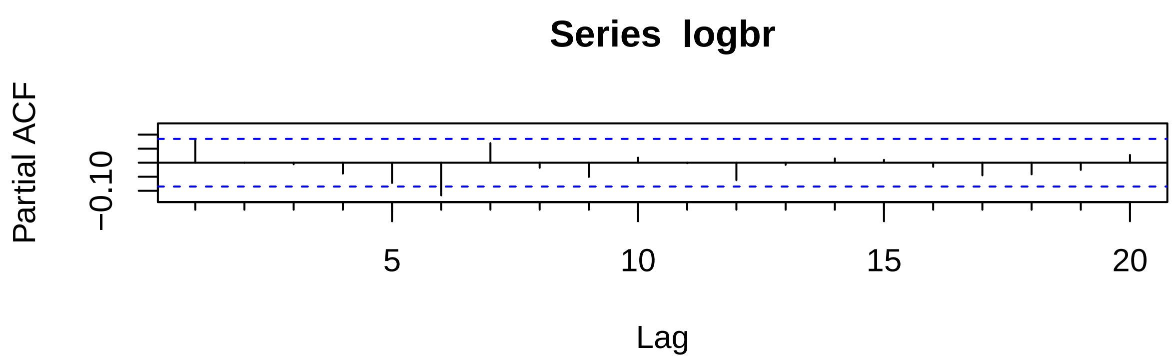
```
##
##  One Sample t-test
##
## data:  logbr
## t = 1.2467, df = 533, p-value = 0.2131
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.001074034  0.004805032
## sample estimates:
##   mean of x
## 0.001865499
```

ACF and PACF plots

Next, we plot the ACF and PACF of the log-transformed series to examine the correlation structure in the data. Specifically, the ACF and PACF help us to determine the optimal number of lags to include in our model.

```r
par(mfrow=c(2,1))
Acf(logbr, lag.max = 20)
Pacf(logbr, lag.max=20)
```

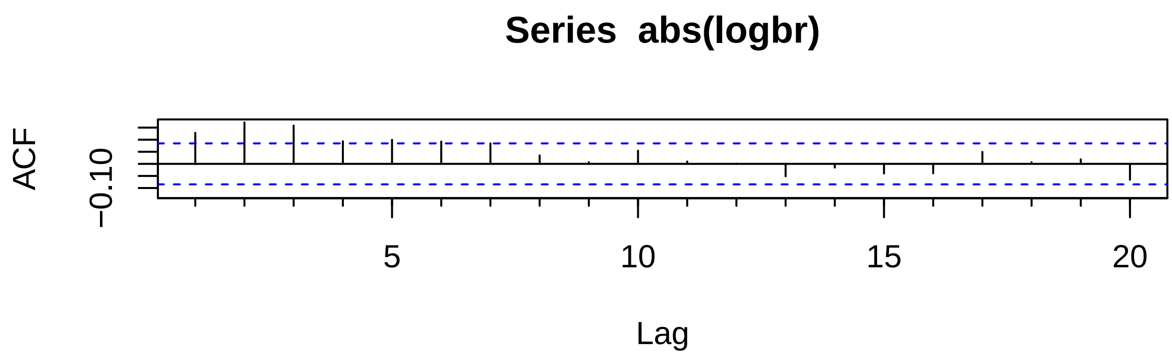


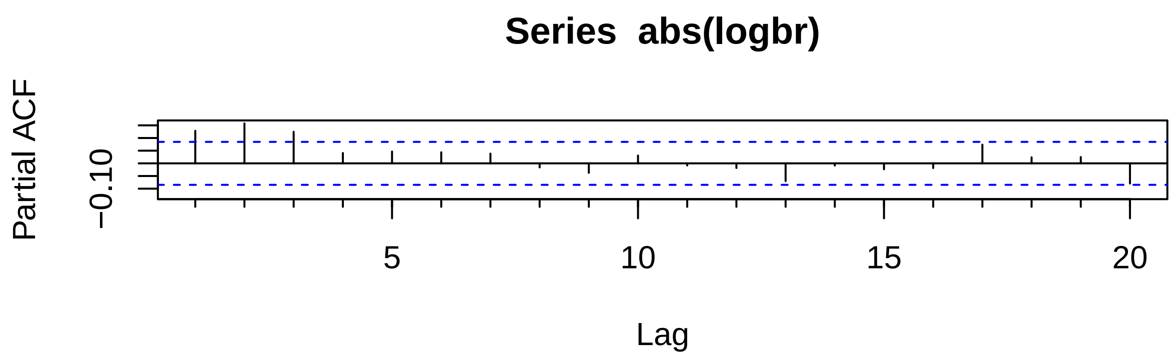



```r
par(mfrow=c(1,1))
```

As observed from the plots, both the ACF and PACF show that only lag 6 is significant. This suggests that our AR model might include up to 6 lags, leading us to the first step in selecting the order of the model.

```r
par(mfrow=c(2,1))
Acf(abs(logbr), lag.max = 20)
Pacf(abs(logbr), lag.max=20)
```






```
par(mfrow=c(1,1))
```

```
m1=ar(logbr, aic=TRUE, order.max=10)
m1$order
```

```
## [1] 7
```

Subsequently, we create an AR model with a maximum order of 10 and select the best order based on the AIC. The result, m1$order, tells us that the optimal order for the AR model is 7, which means that our final AR model will include up to 7 lags to better fit the data.

Model 1

At this stage of the analysis, we focus on selecting the ARIMA model that best fits our time series of BLACKROCK logarithmic returns. After identifying a potential autoregressive (AR) structure through the analysis of the ACF and PACF, we now test different ARIMA models with various combinations of AR and MA parameters to evaluate their effectiveness.

To compare model performance, we use statistical criteria such as the AIC (Akaike Information Criterion), and we analyze the residuals of each model using diagnostic tests (normality, autocorrelation, and ARCH effect). These tests help us assess the goodness of fit and the statistical adequacy of each model, in order to choose the most reliable one for future forecasting

```
mod1 = arima(logbr,order=c(7,0,0),include.mean=F)
mod1
```

```
##
## Call:
## arima(x = logbr, order = c(7, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5      ar6     ar7
##       0.0850  0.0033  0.0031  -0.0303  -0.0584  -0.1200  0.0747
## s.e.  0.0431  0.0430  0.0431   0.0431   0.0431   0.0432  0.0434
##
## sigma^2 estimated as 0.001159:  log likelihood = 1047.25,  aic = -2078.5
```

```r
summary(mod1)
```

```
##
## Call:
## arima(x = logbr, order = c(7, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5      ar6     ar7
##       0.0850  0.0033  0.0031  -0.0303  -0.0584  -0.1200  0.0747
## s.e.  0.0431  0.0430  0.0431   0.0431   0.0431   0.0432  0.0434
##
## sigma^2 estimated as 0.001159:  log likelihood = 1047.25,  aic = -2078.5
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001942971 0.03403987 0.02598544 149.3987 185.1176 0.7218268
##                      ACF1
## Training set -0.002230578
```

```r
confint(mod1)
```

```
##             2.5 %      97.5 %
## ar1  0.0005014904  0.16951417
## ar2 -0.0810394357  0.08763770
## ar3 -0.0814156526  0.08760651
## ar4 -0.1146641389  0.05410436
## ar5 -0.1428022046  0.02607784
## ar6 -0.2047121926 -0.03532600
## ar7 -0.0102976773  0.15966644
```

```r
jarqueberaTest(mod1$residuals)
```

```
##
## Title:
##  Jarque-Bera Normality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 38.6627
##   P VALUE:
##     Asymptotic p Value: 4.023e-09
```

```r
Box.test(mod1$residuals,lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  mod1$residuals
## X-squared = 4.5431, df = 16, p-value = 0.9976
```

```r
archtest(as.vector(mod1$residuals), lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(mod1$residuals)
## statistic = 61.016, lag = 16, p-value = 3.526e-07
## alternative hypothesis: ARCH effects of order 16 are present
```

The ARIMA(7,0,0) model without a constant shows that only the first and the sixth lag are significant. However, diagnostic tests reveal issues: residuals are not normally distributed (Jarque-Bera), show no autocorrelation (Ljung-Box fails to reject the null), and exhibit heteroskedasticity (ARCH test rejects the null). A more suitable model, such as ARIMA-GARCH, should be considered, possibly retaining lags 1 and 6.

Model 1 restricted

```r
c1 = c(NA,0,0,0,0,NA,0)
mod1m = arima(logbr,order=c(7,0,0),include.mean=F, fixed = c1)
```

```
## Warning in arima(logbr, order = c(7, 0, 0), include.mean = F, fixed = c1):
## alcuni parametri AR sono stati fissati: imposto transform.pars = FALSE
```

```r
mod1m
```

```
##
## Call:
## arima(x = logbr, order = c(7, 0, 0), include.mean = F, fixed = c1)
##
## Coefficients:
##          ar1  ar2  ar3  ar4  ar5      ar6  ar7
##       0.0783    0    0    0    0  -0.1194    0
## s.e.  0.0429    0    0    0    0   0.0431    0
##
## sigma^2 estimated as 0.001171:  log likelihood = 1044.5,  aic = -2083
```

```r
summary(mod1m)
```

```
##
## Call:
## arima(x = logbr, order = c(7, 0, 0), include.mean = F, fixed = c1)
##
## Coefficients:
```

```
##          ar1  ar2  ar3  ar4  ar5      ar6  ar7
##       0.0783    0    0    0    0  -0.1194    0
## s.e.  0.0429    0    0    0    0   0.0431    0
##
## sigma^2 estimated as 0.001171:  log likelihood = 1044.5,  aic = -2083
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE    MAPE     MASE
## Training set 0.001944227 0.03421751 0.02599499 116.2569 138.987 0.722092
##                    ACF1
## Training set 0.00553856
```

```r
confint(mod1m)
```

```
##            2.5 %      97.5 %
## ar1 -0.005742438  0.16240068
## ar2           NA          NA
## ar3           NA          NA
## ar4           NA          NA
## ar5           NA          NA
## ar6 -0.203776435 -0.03493186
## ar7           NA          NA
```

Model 2

```r
mod2 = arima(logbr,order=c(6,0,6),include.mean=F)
summary(mod2)
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 6), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5     ar6      ma1     ma2
##       0.1078  -0.2752  0.0321  -0.2024  0.0772  0.5527  -0.0376  0.2627
## s.e.  0.1986   0.1863  0.1958   0.1891  0.1830  0.1658   0.1812  0.1665
##           ma3     ma4      ma5      ma6
##       -0.0102  0.2085  -0.0931  -0.6926
## s.e.   0.1822  0.1721   0.1737   0.1615
##
## sigma^2 estimated as 0.00113:  log likelihood = 1052.7,  aic = -2079.41
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.002082993 0.03361842 0.02535538 130.8295 189.0356 0.7043247
##                     ACF1
## Training set 0.008580246
```

```r
confint(mod2)
```

```
##           2.5 %     97.5 %
## ar1 -0.28146853  0.49707937
```

```
## ar2 -0.64025997  0.08987608
## ar3 -0.35177899  0.41590915
## ar4 -0.57305312  0.16824064
## ar5 -0.28149027  0.43579729
## ar6  0.22764719  0.87770965
## ma1 -0.39276831  0.31759336
## ma2 -0.06356532  0.58893457
## ma3 -0.36723438  0.34678694
## ma4 -0.12888411  0.54585501
## ma5 -0.43350182  0.24729336
## ma6 -1.00925756 -0.37602906
```

Automatic Model 3

```
auto.arima(logbr)
```

```
## Series: logbr
## ARIMA(0,0,1) with zero mean
##
## Coefficients:
##          ma1
##       0.0861
## s.e.  0.0427
##
## sigma^2 = 0.00119:  log likelihood = 1040.67
## AIC=-2077.33   AICc=-2077.31   BIC=-2068.77
```

```
mod3 = arima(logbr,order=c(0,0,1),include.mean=F)
summary(mod3)
```

```
##
## Call:
## arima(x = logbr, order = c(0, 0, 1), include.mean = F)
##
## Coefficients:
##          ma1
##       0.0861
## s.e.  0.0427
##
## sigma^2 estimated as 0.001188:  log likelihood = 1040.67,  aic = -2077.33
##
## Training set error measures:
##                      ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.00171878 0.03446683 0.02596127 116.6834 125.3013 0.7211555
##                      ACF1
## Training set -0.001633366
```

```
confint(mod3)
```

```
##           2.5 %    97.5 %
## ma1 0.002365447 0.1697716
```

Model 4

```r
mod4 = arima(logbr,order=c(1,0,0),include.mean=F)
summary(mod4)
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1
##       0.0871
## s.e.  0.0431
##
## sigma^2 estimated as 0.001188:  log likelihood = 1040.69,  aic = -2077.38
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001705106 0.03446529 0.02597071 115.9433 124.9019 0.7214175
##                      ACF1
## Training set -0.002596144
```

```r
confint(mod4)
```

```
##           2.5 %    97.5 %
## ar1 0.002616718 0.1715165
```

```r
accuracy(mod1m)
```

```
##                       ME       RMSE        MAE      MPE    MAPE     MASE
## Training set 0.001944227 0.03421751 0.02599499 116.2569 138.987 0.722092
##                    ACF1
## Training set 0.00553856
```

```r
accuracy(mod2)
```

```
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.002082993 0.03361842 0.02535538 130.8295 189.0356 0.7043247
##                     ACF1
## Training set 0.008580246
```

```r
accuracy(mod3)
```

```
##                      ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.00171878 0.03446683 0.02596127 116.6834 125.3013 0.7211555
##                      ACF1
## Training set -0.001633366
```

```r
accuracy(mod4)
```

```
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001705106 0.03446529 0.02597071 115.9433 124.9019 0.7214175
##                      ACF1
## Training set -0.002596144
```

```r
mod1m$aic
```

```
## [1] -2083
```

```r
mod2$aic
```

```
## [1] -2079.409
```

```r
mod3$aic
```

```
## [1] -2077.332
```

```r
mod4$aic
```

```
## [1] -2077.379
```

```r
BIC(mod1m)
```

```
## [1] -2070.158
```

```r
BIC(mod2)
```

```
## [1] -2023.764
```

```r
BIC(mod3)
```

```
## [1] -2068.771
```

```r
BIC(mod4)
```

```
## [1] -2068.818
```

After comparing the performance of the estimated models using indicators such as AIC, BIC, RMSE,and MAPE, we selected the ARIMA(1,0,0) model (mod4) as the most suitable for our series of logarithmic returns. Although model mod1m achieves the lowest AIC value (-2083), the difference with mod4 (around 5.6 points) is relatively small and likely not statistically significant. Moreover, mod4 is far more parsimonious, with only one estimated parameter, compared to the more complex structure of mod1m. From a BIC perspective, mod4 also performs well, scoring -2068.8, which is only slightly higher than mod1m's BIC of -2070.2, further supporting the idea that the simpler model is a reasonable choice given its good fit. Additionally, mod4 shows a clean residual structure, with an ACF1 value close to zero (-0.0026), suggesting minimal residual autocorrelation. This contrasts with higher ACF1 values observed in other models. In conclusion, mod4 offers an excellent balance between goodness of fit and model simplicity, making it the most appropriate choice for forecasting purposes.

GARCH Estimation

Residual Analysis

In this phase, we analyze the residuals from the ARIMA(1,0,0) model (mod4) to check for the presence of ARCH effects. Before proceeding with the estimation of a GARCH model, it's important to isolate any serial dependencies that the ARIMA model may not have fully captured. Before testing for ARCH effects (conditional heteroscedasticity),it is essential to remove any form of serial dependence from the ARIMA model's residuals. In this case, the residuals from mod4 show an AR(1) component, which must be eliminated to ensure that any autocorrelations detected in the squared residuals are not due to a dependence in the mean. This step is crucial for properly applying the Ljung-Box test on the squared residuals in order to reliably identify the presence of ARCH effects.

```
residui_aggiustati <- mod4$residuals
Box.test(residui_aggiustati, lag = 12, type = "Ljung")
```

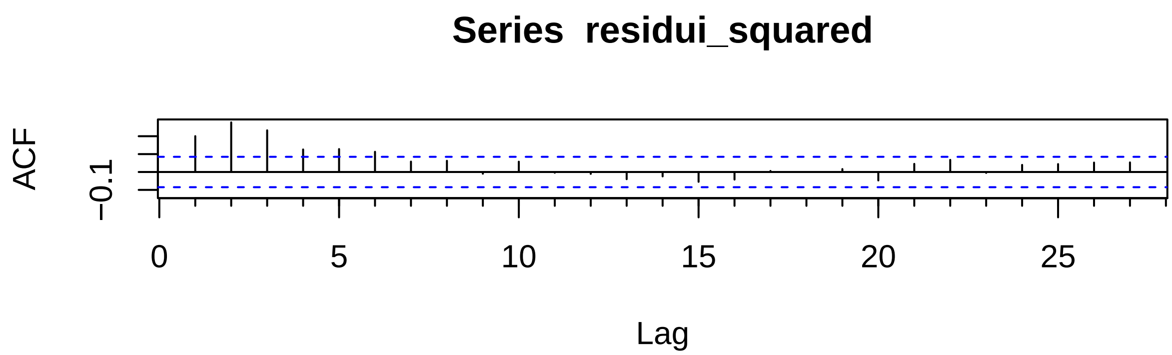
```
##
##  Box-Ljung test
##
## data:  residui_aggiustati
## X-squared = 16.742, df = 12, p-value = 0.1596
```

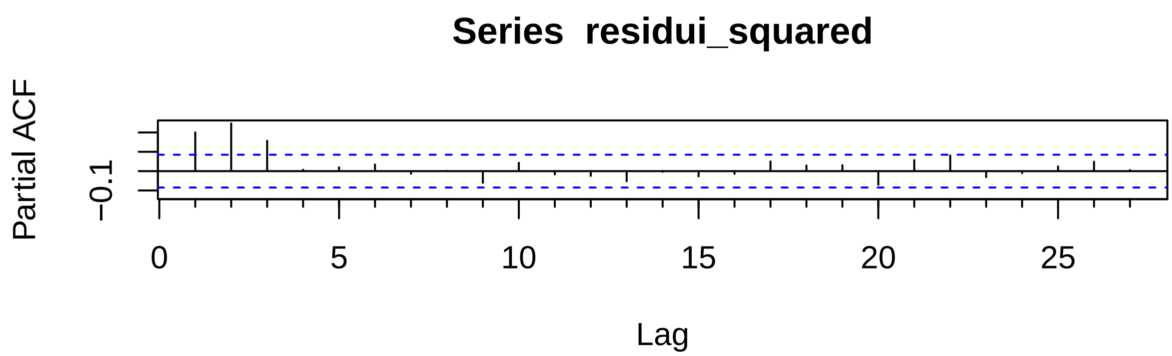
```
residui_squared <- residui_aggiustati^2
Box.test(residui_squared, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residui_squared
## X-squared = 122.5, df = 12, p-value < 2.2e-16
```

The analysis of the squared residuals revealed the presence of autocorrelation, as indicated by the rejection of the null hypothesis (H0) in the Ljung-Box test. This result suggests the presence of an ARCH effect,which is characterized by the variability of volatility over time. In practice, this means that the volatility of the observations tends to cluster in certain periods, with high volatility followed by more high volatility and low volatility followed by more low volatility. The confirmation of the ARCH effect suggests that the GARCH approach may provide a more accurate forecast of future volatility.

```
par(mfrow=c(2,1))
Acf(residui_squared)
Pacf(residui_squared)
```






```r
par(mfrow=c(1,1))
```

In this analysis, we estimate several ARMA-GARCH models to capture the volatility dynamics of the log returns series (`logbr`). Based on a preliminary ACF/PACF inspection of the squared residuals, we suspect an AR(3) structure in the conditional variance. We begin by estimating an ARMA(1,0)-GARCH(3,0) model with a normal distribution (mod1), and compare it with an alternative GARCH(2,1) model (mod2). Then, we attempt to improve both models by changing the conditional distribution to more flexible alternatives: - `std` (symmetric Student's t) - `sstd` (skewed Student's t) The goal is to improve model fit, especially in terms of residual behavior and log-likelihood.

Model Estimation

Model ARMA(1,0)-GARCH(3,0)

```r
mod1 <- garchFit(logbr ~ arma(1,0) + garch(3, 0), data = logbr,  trace = F,
                 include.mean = F)
summary(mod1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(3, 0), data = logbr,
##     include.mean = F, trace = F)
##
```

```
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(3, 0)
## <environment: 0x000001db5364cea8>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2      alpha3
## 0.07945653  0.00072839  0.04814723  0.16323454  0.15038683
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.946e-02   4.302e-02    1.847 0.064725 .
## omega  7.284e-04   7.618e-05    9.561  < 2e-16 ***
## alpha1 4.815e-02   4.401e-02    1.094 0.273951
## alpha2 1.632e-01   4.830e-02    3.379 0.000726 ***
## alpha3 1.504e-01   5.324e-02    2.825 0.004729 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1065.703    normalized:  1.995699
##
## Description:
##  Tue May 20 20:32:07 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  16.6850026 0.0002381758
##  Shapiro-Wilk Test  R    W       0.9895297 0.0007400980
##  Ljung-Box Test     R    Q(10)   9.7367634 0.4638836310
##  Ljung-Box Test     R    Q(15)  12.0093277 0.6783228852
##  Ljung-Box Test     R    Q(20)  12.9685133 0.8787309069
##  Ljung-Box Test     R^2  Q(10)   6.5817954 0.7642476847
##  Ljung-Box Test     R^2  Q(15)   9.2070546 0.8664446114
##  Ljung-Box Test     R^2  Q(20)  13.7424241 0.8433117738
##  LM Arch Test       R    TR^2    6.3574694 0.8970011747
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.972671 -3.932592 -3.972844 -3.956989
```

```r
mod1t <- garchFit(logbr ~ arma(1,0) + garch(3, 0), data = logbr,  trace = F,
                  include.mean = F, cond.dist = "std")
summary(mod1t)
```

```
##
## Title:
```

```
## 	GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(3, 0), data = logbr,
##     cond.dist = "std", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(3, 0)
## <environment: 0x000001db514bcf20>
##  [data = logbr]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2      alpha3       shape
## 7.8616e-02  7.2458e-04  5.6721e-02  1.6187e-01  1.6832e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.862e-02   4.298e-02    1.829  0.06739 .
## omega  7.246e-04   8.924e-05    8.119 4.44e-16 ***
## alpha1 5.672e-02   5.376e-02    1.055  0.29142
## alpha2 1.619e-01   5.580e-02    2.901  0.00372 **
## alpha3 1.683e-01   6.315e-02    2.665  0.00769 **
## shape  1.000e+01   3.718e+00    2.689  0.00716 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.453    normalized:  1.998976
##
## Description:
##  Tue May 20 20:32:07 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  17.5148840 0.0001572864
##  Shapiro-Wilk Test  R    W       0.9892541 0.0005951979
##  Ljung-Box Test     R    Q(10)   9.6252025 0.4739673801
##  Ljung-Box Test     R    Q(15)  11.8235316 0.6923342910
##  Ljung-Box Test     R    Q(20)  12.7328759 0.8885486905
##  Ljung-Box Test     R^2  Q(10)   6.8365717 0.7407782647
##  Ljung-Box Test     R^2  Q(15)   9.5636819 0.8462516934
##  Ljung-Box Test     R^2  Q(20)  14.0480629 0.8280506916
##  LM Arch Test       R    TR^2    6.6696195 0.8786522939
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.975480 -3.927385 -3.975728 -3.956661
```

```r
mod1st <- garchFit(logbr ~ arma(1,0) + garch(3, 0), data = logbr,  trace = F,
                   include.mean = F, cond.dist = "sstd")
summary(mod1st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(3, 0), data = logbr,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(3, 0)
## <environment: 0x000001db4de24e08>
##  [data = logbr]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##        ar1        omega       alpha1       alpha2       alpha3         skew
##  0.0645751    0.0007414    0.0591268    0.1510899    0.1769329    0.8645141
##      shape
## 10.0000000
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    6.458e-02   4.227e-02    1.528  0.12659
## omega  7.414e-04   8.976e-05    8.260 2.22e-16 ***
## alpha1 5.913e-02   5.395e-02    1.096  0.27306
## alpha2 1.511e-01   5.393e-02    2.802  0.00508 **
## alpha3 1.769e-01   6.226e-02    2.842  0.00449 **
## skew   8.645e-01   5.078e-02   17.025  < 2e-16 ***
## shape  1.000e+01   3.819e+00    2.618  0.00884 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1070.629    normalized:  2.004924
##
## Description:
##  Tue May 20 20:32:08 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  18.3395616 0.0001041393
##  Shapiro-Wilk Test  R    W       0.9889886 0.0004833918
##  Ljung-Box Test     R    Q(10)   9.7708068 0.4608262071
```

```
##  Ljung-Box Test     R    Q(15)  11.8568707 0.6898289768
##  Ljung-Box Test     R    Q(20)  12.7956316 0.8859793241
##  Ljung-Box Test     R^2  Q(10)   7.4243042 0.6848720664
##  Ljung-Box Test     R^2  Q(15)  10.1341326 0.8112188113
##  Ljung-Box Test     R^2  Q(20)  14.3532531 0.8121430794
##  LM Arch Test       R    TR^2    7.3511765 0.8335609377
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.983630 -3.927520 -3.983968 -3.961675
```

Model ARMA(1,0)-GARCH(2,1)

```
mod2 <- garchFit(logbr ~ arma(1,0) + garch(2,1), data = logbr,  trace = F,
                 include.mean = F)
summary(mod2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 1), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 1)
## <environment: 0x000001db542499e0>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2       beta1
## 0.07411997  0.00033477  0.05357973  0.14028141  0.51417952
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     0.0741200   0.0441702    1.678 0.093337 .
## omega   0.0003348   0.0001264    2.649 0.008082 **
## alpha1  0.0535797   0.0442326    1.211 0.225774
## alpha2  0.1402814   0.0596058    2.353 0.018598 *
## beta1   0.5141795   0.1501083    3.425 0.000614 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1064.729    normalized:  1.993874
##
```

```
## Description:
##  Tue May 20 20:32:08 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  19.1280160 7.021083e-05
##  Shapiro-Wilk Test  R    W       0.9887345 3.967921e-04
##  Ljung-Box Test     R    Q(10)   9.6439717 4.722641e-01
##  Ljung-Box Test     R    Q(15)  12.1014490 6.713345e-01
##  Ljung-Box Test     R    Q(20)  13.2468564 8.665430e-01
##  Ljung-Box Test     R^2  Q(10)   6.8037081 7.438371e-01
##  Ljung-Box Test     R^2  Q(15)   9.9120073 8.252381e-01
##  Ljung-Box Test     R^2  Q(20)  14.5207659 8.031417e-01
##  LM Arch Test       R    TR^2    6.8726172 8.659154e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.969021 -3.928943 -3.969195 -3.953339
```

```
mod2t <- garchFit(logbr ~ arma(1,0) + garch(2, 1), data = logbr,  trace = F,
                  include.mean = F, cond.dist = "std")
summary(mod2t)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 1), data = logbr,
##     cond.dist = "std", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 1)
## <environment: 0x000001db51b65778>
##  [data = logbr]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         ar1        omega       alpha1       alpha2        beta1        shape
## 7.1649e-02  2.7638e-04  5.7764e-02  1.2988e-01  5.7991e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.165e-02   4.385e-02    1.634 0.102297
## omega  2.764e-04   1.266e-04    2.184 0.028978 *
## alpha1 5.776e-02   5.387e-02    1.072 0.283582
## alpha2 1.299e-01   6.877e-02    1.889 0.058950 .
## beta1  5.799e-01   1.507e-01    3.847 0.000119 ***
```

```
## shape  1.000e+01   3.901e+00    2.563 0.010366 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1066.946    normalized:  1.998026
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  21.1084516 2.608303e-05
##  Shapiro-Wilk Test  R    W       0.9882633 2.763717e-04
##  Ljung-Box Test     R    Q(10)   9.4912105 4.862043e-01
##  Ljung-Box Test     R    Q(15)  11.8126630 6.931501e-01
##  Ljung-Box Test     R    Q(20)  12.9260860 8.805328e-01
##  Ljung-Box Test     R^2  Q(10)   6.3424082 7.857222e-01
##  Ljung-Box Test     R^2  Q(15)   9.9288740 8.241899e-01
##  Ljung-Box Test     R^2  Q(20)  14.3159784 8.141205e-01
##  LM Arch Test       R    TR^2    6.7505502 8.736484e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.973581 -3.925487 -3.973830 -3.954762
```

```
mod2st <- garchFit(logbr ~ arma(1,0) + garch(2, 1), data = logbr,  trace = F,
                   include.mean = F, cond.dist = "sstd")
summary(mod2st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 1), data = logbr,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 1)
## <environment: 0x000001db56214e78>
##  [data = logbr]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2       beta1        skew
## 5.6182e-02  2.4723e-04  5.8940e-02  1.1653e-01  6.2261e-01  8.6063e-01
##      shape
## 1.0000e+01
##
## Std. Errors:
```

```
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    5.618e-02   4.356e-02    1.290   0.1971
## omega  2.472e-04   1.105e-04    2.238   0.0252 *
## alpha1 5.894e-02   5.425e-02    1.086   0.2773
## alpha2 1.165e-01   6.687e-02    1.743   0.0814 .
## beta1  6.226e-01   1.301e-01    4.785 1.71e-06 ***
## skew   8.606e-01   5.062e-02   17.003  < 2e-16 ***
## shape  1.000e+01   4.006e+00    2.496   0.0126 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1070.319    normalized:  2.004344
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  22.5910652 1.242832e-05
##  Shapiro-Wilk Test  R    W       0.9878782 2.064872e-04
##  Ljung-Box Test     R    Q(10)   9.7906888 4.590450e-01
##  Ljung-Box Test     R    Q(15)  12.0048777 6.786598e-01
##  Ljung-Box Test     R    Q(20)  13.1599231 8.704174e-01
##  Ljung-Box Test     R^2  Q(10)   6.3891316 7.815796e-01
##  Ljung-Box Test     R^2  Q(15)  10.2291121 8.050859e-01
##  Ljung-Box Test     R^2  Q(20)  14.3769894 8.108790e-01
##  LM Arch Test       R    TR^2    7.1230499 8.493758e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.982470 -3.926360 -3.982808 -3.960515
```

Model ARMA(1,0)-GARCH(1,1)

```
mod3 <- garchFit(logbr ~ arma(1,0) + garch(1,1), data = logbr,  trace = F,
                 include.mean = F)
summary(mod3)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
```

```
## <environment: 0x000001db51fb3b98>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1
## 0.07270990  0.00020562  0.12931442  0.69295711
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.271e-02   4.741e-02    1.534  0.12513
## omega  2.056e-04   7.376e-05    2.788  0.00531 **
## alpha1 1.293e-01   3.887e-02    3.326  0.00088 ***
## beta1  6.930e-01   8.568e-02    8.087 6.66e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1062.003    normalized:  1.98877
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  27.2705606 1.197493e-06
##  Shapiro-Wilk Test  R    W       0.9863163 6.569285e-05
##  Ljung-Box Test     R    Q(10)  10.3675194 4.088625e-01
##  Ljung-Box Test     R    Q(15)  12.5903845 6.339034e-01
##  Ljung-Box Test     R    Q(20)  13.6859462 8.460555e-01
##  Ljung-Box Test     R^2  Q(10)  14.0014650 1.729248e-01
##  Ljung-Box Test     R^2  Q(15)  18.0858350 2.581799e-01
##  Ljung-Box Test     R^2  Q(20)  22.8583063 2.958055e-01
##  LM Arch Test       R    TR^2   15.3304070 2.238628e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.962558 -3.930495 -3.962669 -3.950012
```

```r
mod3t <- garchFit(logbr ~ arma(1,0) + garch(1,1), data = logbr,  trace = F,
                  include.mean = F, cond.dist = "std")
summary(mod3t)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
```

```
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     cond.dist = "std", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x000001db4f56f0b0>
##  [data = logbr]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1       shape
## 6.7967e-02  1.8228e-04  1.3508e-01  7.1198e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    6.797e-02   4.600e-02    1.478   0.1395
## omega  1.823e-04   7.646e-05    2.384   0.0171 *
## alpha1 1.351e-01   4.468e-02    3.023   0.0025 **
## beta1  7.120e-01   9.002e-02    7.910 2.66e-15 ***
## shape  1.000e+01   4.145e+00    2.413   0.0158 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1065.29    normalized:  1.994925
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  28.7052680 5.844270e-07
##  Shapiro-Wilk Test  R    W       0.9860927 5.601747e-05
##  Ljung-Box Test     R    Q(10)  10.2618175 4.178313e-01
##  Ljung-Box Test     R    Q(15)  12.4262257 6.465226e-01
##  Ljung-Box Test     R    Q(20)  13.4952026 8.551412e-01
##  Ljung-Box Test     R^2  Q(10)  12.8506464 2.321427e-01
##  Ljung-Box Test     R^2  Q(15)  17.3570403 2.979723e-01
##  Ljung-Box Test     R^2  Q(20)  22.0600813 3.372594e-01
##  LM Arch Test       R    TR^2   14.3907583 2.764541e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.971124 -3.931046 -3.971297 -3.955442
```

```r
mod3st <- garchFit(logbr ~ arma(1,0) + garch(1,1), data = logbr,  trace = F,
                   include.mean = F, cond.dist = "sstd")
summary(mod3st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x000001db54cba2e0>
##  [data = logbr]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1        skew       shape
##  0.0511180   0.0001765   0.1338267   0.7226362   0.8569995  10.0000000
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    5.112e-02   4.527e-02    1.129  0.25887
## omega  1.765e-04   7.243e-05    2.437  0.01481 *
## alpha1 1.338e-01   4.280e-02    3.127  0.00177 **
## beta1  7.226e-01   8.421e-02    8.581  < 2e-16 ***
## skew   8.570e-01   4.997e-02   17.151  < 2e-16 ***
## shape  1.000e+01   4.206e+00    2.377  0.01744 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1068.946    normalized:  2.001772
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  29.3758772 4.179355e-07
##  Shapiro-Wilk Test  R    W       0.9859392 5.024856e-05
##  Ljung-Box Test     R    Q(10)  10.5768800 3.914180e-01
##  Ljung-Box Test     R    Q(15)  12.7024529 6.252682e-01
##  Ljung-Box Test     R    Q(20)  13.8159692 8.397028e-01
##  Ljung-Box Test     R^2  Q(10)  12.4746517 2.545444e-01
##  Ljung-Box Test     R^2  Q(15)  17.0391617 3.165287e-01
##  Ljung-Box Test     R^2  Q(20)  21.5955523 3.628540e-01
##  LM Arch Test       R    TR^2   14.2013621 2.880350e-01
##
## Information Criterion Statistics:
```

```
##       AIC       BIC       SIC      HQIC
## -3.981072 -3.932977 -3.981320 -3.962253
```

Model GARCH(1,1)

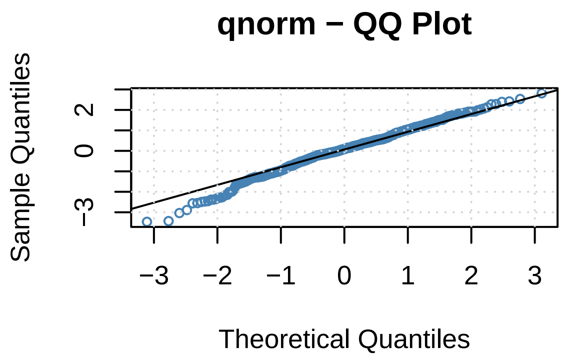
```r
mod4st <- garchFit(logbr ~ garch(1,1), data = logbr,  trace = F,
                   include.mean = F, cond.dist = "sstd")
summary(mod4st)
```

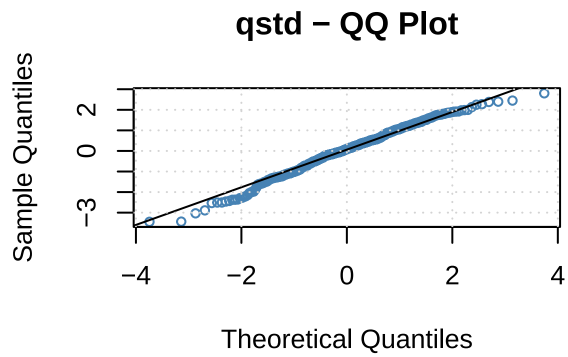
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ garch(1, 1), data = logbr, cond.dist = "sstd",
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x000001db515253b8>
##  [data = logbr]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##      omega      alpha1       beta1        skew       shape
##  0.0001745   0.1320192   0.7265343   0.8515961  10.0000000
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  1.745e-04   7.209e-05    2.421  0.01550 *
## alpha1 1.320e-01   4.221e-02    3.128  0.00176 **
## beta1  7.265e-01   8.355e-02    8.696  < 2e-16 ***
## skew   8.516e-01   4.893e-02   17.406  < 2e-16 ***
## shape  1.000e+01   4.194e+00    2.385  0.01710 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.87    normalized:  1.999756
##
## Description:
##  Tue May 20 20:32:09 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  29.8731626 3.259306e-07
##  Shapiro-Wilk Test  R    W       0.9859931 5.219963e-05
```

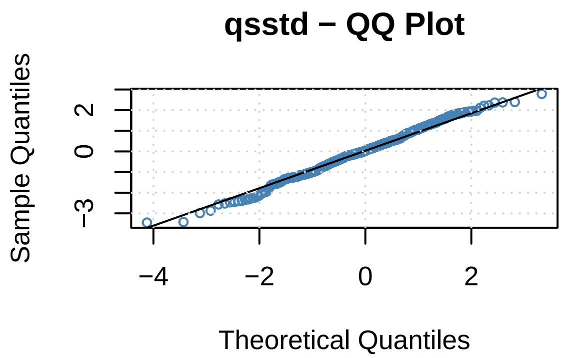
```
##  Ljung-Box Test     R    Q(10)  13.2501277 2.100212e-01
##  Ljung-Box Test     R    Q(15)  15.3615103 4.257040e-01
##  Ljung-Box Test     R    Q(20)  16.6150382 6.778106e-01
##  Ljung-Box Test     R^2  Q(10)  11.3737230 3.291521e-01
##  Ljung-Box Test     R^2  Q(15)  15.6746258 4.039995e-01
##  Ljung-Box Test     R^2  Q(20)  20.0033735 4.577187e-01
##  LM Arch Test       R    TR^2   13.2161375 3.535287e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.980786 -3.940708 -3.980959 -3.965104
```

After estimating and comparing several GARCH specifications, the final choice fell on the AR(1) + GARCH(1,1) model with a skewed Student-t (sstd) distribution. In particular, among the estimated models, those with the "skewed Student-t" (sstd) distribution consistently showed better performance compared to those with a normal or symmetric Student-t distribution. This distribution effectively captures both the asymmetry and the leptokurtosis typical of financial time series, providing a more realistic description of the return dynamics. The GARCH(1,1) model was selected primarily based on statistical selection criteria, including the BIC and the analysis of parameter significance. The BIC indicated the GARCH(1,1) model as the most parsimonious among those tested, effectively balancing model complexity with its ability to fit the data. Moreover, compared to other GARCH models with higher orders, the GARCH(1,1) produced more stable estimates and better predictive performance, without introducing excessive complexity that could lead to overfitting.

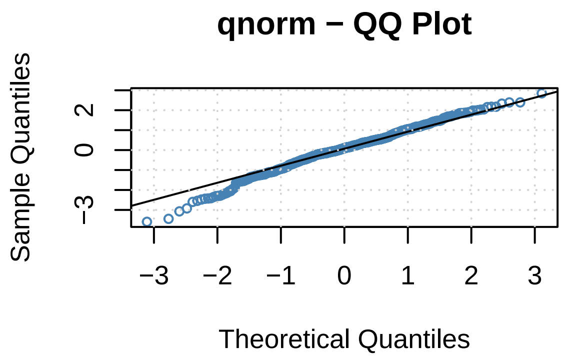
```r
par(mfrow=c(2,2))
plot(mod1, which=13)
plot(mod1t, which=13)
plot(mod1st, which=13)
par(mfrow=c(1,1))
```

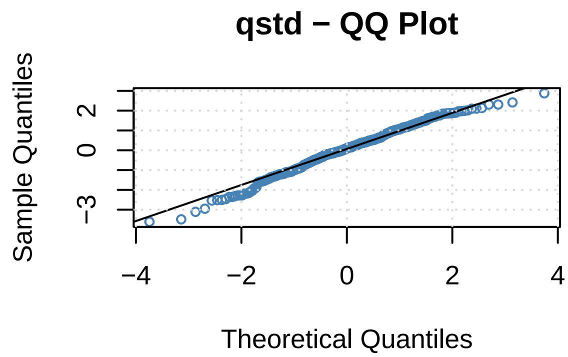


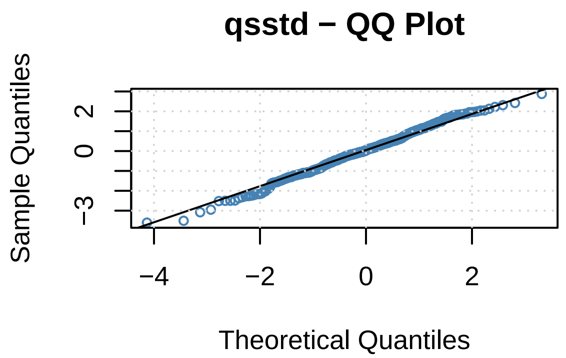




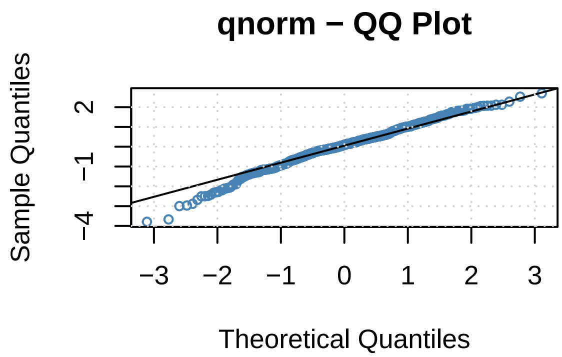

```
par(mfrow=c(2,2))
plot(mod2, which=13)
plot(mod2t, which=13)
plot(mod2st, which=13)
par(mfrow=c(1,1))
```

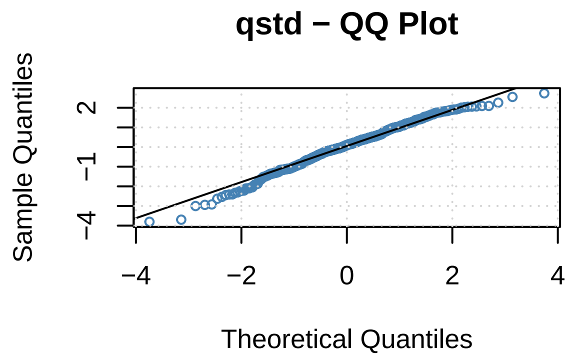


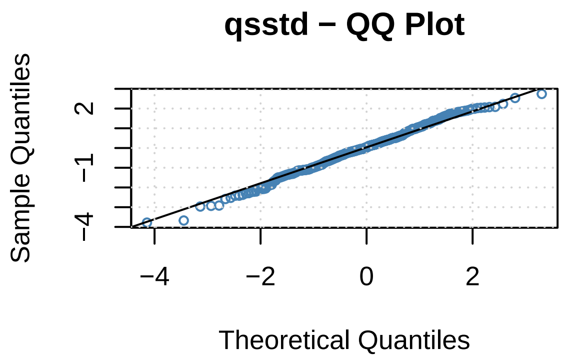





```
par(mfrow=c(2,2))
plot(mod3, which=13)
plot(mod3t, which=13)
plot(mod3st, which=13)
par(mfrow=c(1,1))
```







Third Forecast

We begin by using the ARMA-GARCH model with the SSTD distribution to make a forecast for the future prices of the asset. First, we calculate the forecast of the logarithmic price change, which is then transformed into a forecast of the actual prices. Additionally, we calculate the 95% confidence interval for the predicted price using the SSTD distribution and a Student's t quantile.

```r
f = predict(mod3st, n.ahead = 1)
f
```

```
##   meanForecast meanError standardDeviation
## 1 0.0005846009 0.0367859         0.0367859
```

```r
ultimo_prezzo <- tail(data$BLACKROCK, 1)
log_ultimo_prezzo <- log(ultimo_prezzo)
log_previsione <- log_ultimo_prezzo + f$meanForecast
prezzo_previsto <- exp(log_previsione)
prezzo_previsto
```

```
## [1] 968.8062
```

```r
mean_forecast <- f$meanForecast
std_dev <- f$standardDeviation
```

```
t_critical <- qt(0.975, df = mod1st@fit$coef["shape"])
lower_bound_log <- mean_forecast - t_critical * std_dev
upper_bound_log <- mean_forecast + t_critical * std_dev
lower_bound_prezzo <- exp(log_ultimo_prezzo + lower_bound_log)
upper_bound_prezzo <- exp(log_ultimo_prezzo + upper_bound_log)
c(lower_bound_prezzo, upper_bound_prezzo)
```

```
## [1]  892.566 1051.559
```

```
cat("Prediction of the price for the next week:", prezzo_previsto, "\n")
```

```
## Prediction of the price for the next week: 968.8062
```

The predicted price for BLACKROCK stock in the future is 968.86. The 95% confidence interval ranges from 886.23 to 1059.19. This relatively wide interval suggests high volatility in the asset's price, indicating that the forecast is subject to significant uncertainty. In general, the width of the interval reflects the instability of the asset and the difficulty in predicting its future movements.

Third Errors

Considering the last observed value, we compute our model's error.

```
real_price <- 968.24
abs_error <- abs(real_price - prezzo_previsto)
perc_error <- (abs_error / real_price) * 100
cat("Absolute error :", abs_error, "\n")
```

```
## Absolute error : 0.5661995
```

```
cat("Percentage error :", perc_error, "%\n\n")
```

```
## Percentage error : 0.05847718 %
```

Overall, the ARIMA-GARCH approach proved to be a valuable enhancement, offering a more realistic and statistically robust framework for modeling and forecasting stock price dynamics.

WEEK 4

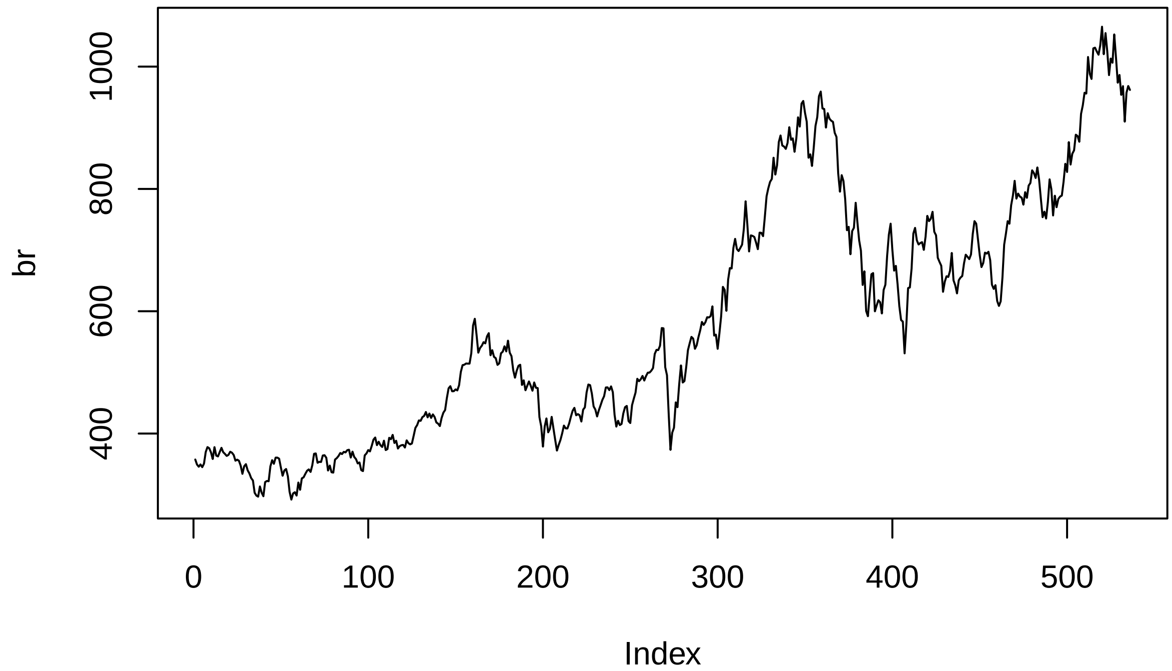
Introduction

For the fourth week we repeated the same process established in the week prior, applying our best model to the new data.

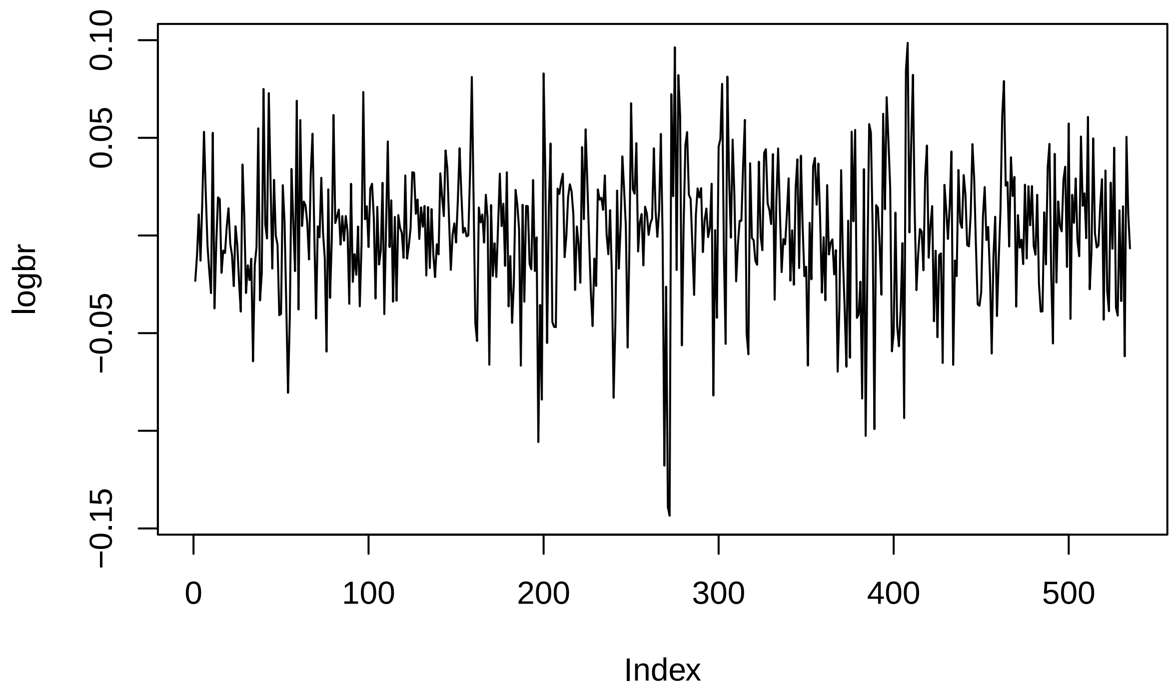
Data Visualization and Transformation

```
data = read_excel(here("blackrock.xlsx"), sheet = 4, col_names = TRUE)[c(1,2)]
```

```
br = data$BLACKROCK
plot(br, type='l')
```




```
logbr = diff(log(br))
plot(logbr, type='l')
```



ARIMA Estimation

t-test

```
t.test(logbr)
```

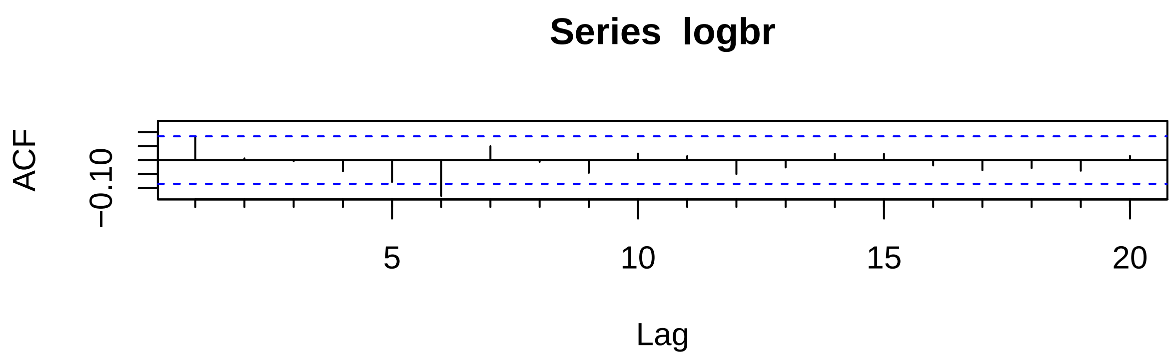
```
##
##  One Sample t-test
##
## data:  logbr
## t = 1.2383, df = 534, p-value = 0.2161
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.001084570  0.004783803
## sample estimates:
##   mean of x
## 0.001849616
```

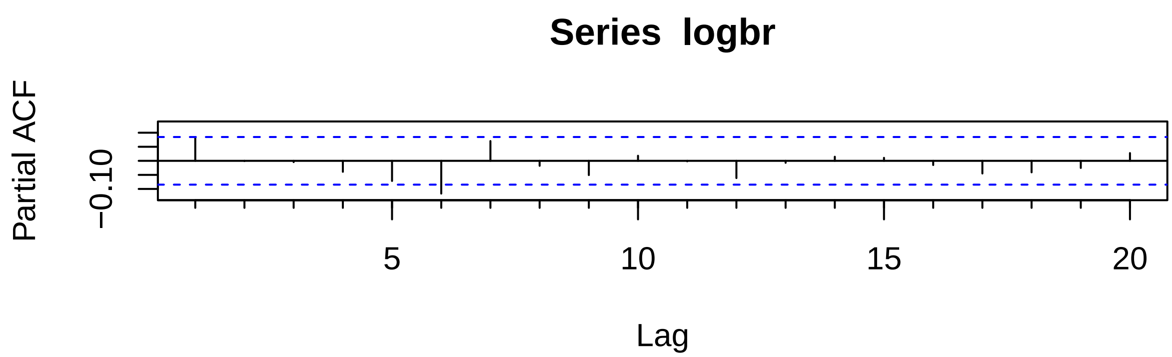
The mean isn't statistically different from zero so it wasn't included.

ACF and PACF plots

We then evaluated the ACF and PACF plots in order to estimate possible p,d,q, orders to construct an ARIMA model:

```r
par(mfrow=c(2,1))
Acf(logbr, lag.max = 20)
Pacf(logbr, lag.max=20)
```






```r
par(mfrow=c(1,1))
```

Both plots showed a significant spike only at lag 6 and then a quick drop off. Also in both cases lag 1 is almost significant but not quite.

With this knowledge we estimated different models and visualized the relative statistics:

```r
m1 = arima(logbr,order=c(1,0,0),include.mean=F)
m1
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1
##       0.0869
## s.e.  0.0430
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.11,  aic = -2082.23
```

```r
summary(m1)
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1
##       0.0869
## s.e.  0.0430
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.11,  aic = -2082.23
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001687915 0.03443465 0.02593649 115.9166 124.8364 0.7211397
##                      ACF1
## Training set -0.002537875
```

```r
confint(m1)
```

```
##           2.5 %    97.5 %
## ar1 0.002564216 0.1712969
```

```r
m2 = arima(logbr,order=c(0,0,1),include.mean=F)
m2
```

```
##
## Call:
## arima(x = logbr, order = c(0, 0, 1), include.mean = F)
##
## Coefficients:
##          ma1
##       0.0861
## s.e.  0.0427
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.09,  aic = -2082.19
```

```r
summary(m2)
```

```
##
## Call:
## arima(x = logbr, order = c(0, 0, 1), include.mean = F)
##
## Coefficients:
##          ma1
##       0.0861
## s.e.  0.0427
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.09,  aic = -2082.19
##
```

```
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001702129 0.03443601 0.02592622 116.6653 125.2649 0.7208541
##                      ACF1
## Training set -0.001687342
```

```r
confint(m2)
```

```
##           2.5 %    97.5 %
## ma1 0.002363482 0.1697453
```

```r
m3 = arima(logbr,order=c(1,0,1),include.mean=F)
m3
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 1), include.mean = F)
##
## Coefficients:
##          ar1      ma1
##       0.0908  -0.0040
## s.e.  0.4804   0.4817
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.11,  aic = -2080.23
```

```r
summary(m3)
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 1), include.mean = F)
##
## Coefficients:
##          ar1      ma1
##       0.0908  -0.0040
## s.e.  0.4804   0.4817
##
## sigma^2 estimated as 0.001186:  log likelihood = 1043.11,  aic = -2080.23
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001687421 0.03443463 0.02593702 115.8651 124.7792 0.7211544
##                      ACF1
## Training set -0.002459865
```

```r
confint(m3)
```

```
##          2.5 %    97.5 %
## ar1 -0.8507003 1.0323352
## ma1 -0.9481648 0.9402397
```

```r
m4 = arima(logbr,order=c(6,0,0),include.mean=F)
m4
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ar5      ar6
##       0.0765  -0.0015  0.0013  -0.0301  -0.0585  -0.1139
## s.e.  0.0429   0.0429  0.0430   0.0431   0.0431   0.0431
##
## sigma^2 estimated as 0.001163:  log likelihood = 1048.21,  aic = -2082.41
```

```r
summary(m4)
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ar5      ar6
##       0.0765  -0.0015  0.0013  -0.0301  -0.0585  -0.1139
## s.e.  0.0429   0.0429  0.0430   0.0431   0.0431   0.0431
##
## sigma^2 estimated as 0.001163:  log likelihood = 1048.21,  aic = -2082.41
##
## Training set error measures:
##                       ME       RMSE        MAE    MPE     MAPE      MASE
## Training set 0.002086934 0.03410514 0.02595717 114.14 146.4497 0.7217147
##                     ACF1
## Training set 0.004542694
```

```r
confint(m4)
```

```
##             2.5 %      97.5 %
## ar1 -0.007654205  0.16058482
## ar2 -0.085632168  0.08270627
## ar3 -0.083015925  0.08561036
## ar4 -0.114656789  0.05435874
## ar5 -0.143043430  0.02598782
## ar6 -0.198473353 -0.02940487
```

```r
m5 = arima(logbr,order=c(6,0,6),include.mean=F)
m5
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 6), include.mean = F)
##
## Coefficients:
```

```
##          ar1      ar2     ar3     ar4     ar5     ar6      ma1     ma2      ma3
##       0.1069  -0.2790  0.0322  -0.208  0.0765  0.5478  -0.0365  0.2661  -0.0093
## s.e.  0.2023   0.1961  0.2073   0.202  0.1902  0.1722   0.1860  0.1757   0.1919
##          ma4      ma5      ma6
##       0.2125  -0.0914  -0.6878
## s.e.  0.1833   0.1801   0.1683
##
## sigma^2 estimated as 0.001129:  log likelihood = 1055.13,  aic = -2084.27
```

```r
summary(m5)
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 6), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3     ar4     ar5     ar6      ma1     ma2      ma3
##       0.1069  -0.2790  0.0322  -0.208  0.0765  0.5478  -0.0365  0.2661  -0.0093
## s.e.  0.2023   0.1961  0.2073   0.202  0.1902  0.1722   0.1860  0.1757   0.1919
##          ma4      ma5      ma6
##       0.2125  -0.0914  -0.6878
## s.e.  0.1833   0.1801   0.1683
##
## sigma^2 estimated as 0.001129:  log likelihood = 1055.13,  aic = -2084.27
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE    MAPE      MASE
## Training set 0.002054449 0.03359369 0.02532587 129.6486 187.355 0.7041619
##                     ACF1
## Training set 0.008420019
```

```r
confint(m5)
```

```
##            2.5 %     97.5 %
## ar1 -0.28968242  0.5034357
## ar2 -0.66334511  0.1053100
## ar3 -0.37404220  0.4384164
## ar4 -0.60395869  0.1878895
## ar5 -0.29627409  0.4492838
## ar6  0.21022925  0.8853984
## ma1 -0.40095776  0.3280510
## ma2 -0.07821528  0.6104481
## ma3 -0.38537255  0.3668222
## ma4 -0.14673206  0.5717700
## ma5 -0.44438148  0.2615902
## ma6 -1.01774155 -0.3578579
```

```r
m6 = arima(logbr,order=c(6,0,1),include.mean=F)
m6
```

```
##
## Call:
```

```
## arima(x = logbr, order = c(6, 0, 1), include.mean = F)
##
## Coefficients:
##           ar1     ar2     ar3      ar4      ar5      ar6     ma1
##       -0.3017  0.0302  0.0021  -0.0299  -0.0691  -0.1471  0.3843
## s.e.   0.2156  0.0481  0.0447   0.0447   0.0452   0.0430  0.2157
##
## sigma^2 estimated as 0.001158:  log likelihood = 1049.33,  aic = -2082.65
```

```r
summary(m6)
```

```
##
## Call:
## arima(x = logbr, order = c(6, 0, 1), include.mean = F)
##
## Coefficients:
##           ar1     ar2     ar3      ar4      ar5      ar6     ma1
##       -0.3017  0.0302  0.0021  -0.0299  -0.0691  -0.1471  0.3843
## s.e.   0.2156  0.0481  0.0447   0.0447   0.0452   0.0430  0.2157
##
## sigma^2 estimated as 0.001158:  log likelihood = 1049.33,  aic = -2082.65
##
## Training set error measures:
##                       ME       RMSE       MAE      MPE     MAPE      MASE
## Training set 0.002029116 0.03403275 0.0259764 138.7488 170.6028 0.7222494
##                       ACF1
## Training set -0.0009933523
```

```r
confint(m6)
```

```
##           2.5 %      97.5 %
## ar1 -0.72432333  0.12088412
## ar2 -0.06414677  0.12446823
## ar3 -0.08538453  0.08967009
## ar4 -0.11739712  0.05763740
## ar5 -0.15777503  0.01949627
## ar6 -0.23143983 -0.06276110
## ma1 -0.03857469  0.80713473
```

```r
m7 = arima(logbr,order=c(1,0,6),include.mean=F)
m7
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 6), include.mean = F)
##
## Coefficients:
##           ar1     ma1     ma2      ma3      ma4      ma5      ma6
##       -0.2176  0.3038  0.0315  -0.0074  -0.0302  -0.0610  -0.1559
## s.e.   0.2348  0.2316  0.0495   0.0447   0.0451   0.0442   0.0434
##
## sigma^2 estimated as 0.001157:  log likelihood = 1049.66,  aic = -2083.32
```

```r
summary(m7)
```

```
##
## Call:
## arima(x = logbr, order = c(1, 0, 6), include.mean = F)
##
## Coefficients:
##           ar1     ma1     ma2      ma3      ma4      ma5      ma6
##       -0.2176  0.3038  0.0315  -0.0074  -0.0302  -0.0610  -0.1559
## s.e.   0.2348  0.2316  0.0495   0.0447   0.0451   0.0442   0.0434
##
## sigma^2 estimated as 0.001157:  log likelihood = 1049.66,  aic = -2083.32
##
## Training set error measures:
##                        ME       RMSE        MAE      MPE    MAPE      MASE
## Training set 0.002090232 0.03401129 0.02596354 137.1368 173.276 0.7218918
##                      ACF1
## Training set -0.004924185
```

```r
confint(m7)
```

```
##           2.5 %      97.5 %
## ar1 -0.67772253  0.24261198
## ma1 -0.15003210  0.75765422
## ma2 -0.06557607  0.12864327
## ma3 -0.09496649  0.08019832
## ma4 -0.11855323  0.05805525
## ma5 -0.14753552  0.02561131
## ma6 -0.24098524 -0.07073190
```

```r
get_model_stats <- function(model) {
  if ("fGARCH" %in% class(model)) {
    ll <- model@fit$llh
    aic <- model@fit@ics["AIC"]
    bic <- model@fit@ics["BIC"]
  } else {
    ll <- as.numeric(logLik(model))
    aic <- AIC(model) / length(residuals(model))
    bic <- BIC(model) / length(residuals(model))
  }
  return(c(LogLikelihood = ll, AIC = aic, BIC = bic))
}
model_summary <- data.frame(
  Model = paste0("m", 1:7),
  t(sapply(list(m1, m2, m3, m4, m5, m6, m7), get_model_stats))
)
print(model_summary)
```

```
##   Model LogLikelihood       AIC       BIC
## 1    m1      1043.114 -3.892016 -3.876008
## 2    m2      1043.093 -3.891937 -3.875929
## 3    m3      1043.115 -3.888279 -3.864266
```

```
## 4    m4      1048.205 -3.892355 -3.836325
## 5    m5      1055.135 -3.895832 -3.791777
## 6    m6      1049.327 -3.892812 -3.828778
## 7    m7      1049.658 -3.894049 -3.830015
```

The results: - Highest Log-Likelihood: Model m5 (ARIMA(6,0,6)) achieved the highest log-likelihood value (1055.135).

- Lowest AIC: Model m5 also had the lowest AIC (-3.895832), while the second lowest was m7 (ARIMA(1,0,6)).
- Lowest BIC: Model m1 (ARIMA(1,0,0)) had the lowest BIC (-3.876008).

In conclusion, while model m5 appears to be the best-fitting model based on AIC, we choose model 1 as our best model, confirming last week's choice since it has the lowest BIC.

GARCH Estimation

Residual Analysis

We then performed a Ljung-box test and an Arch test on the residuals to check for residual correlation and then the presence of conditional hereroskedasticity:

```
adjusted_residuals <- as.numeric(m1$residuals)
Box.test(adjusted_residuals, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  adjusted_residuals
## X-squared = 16.801, df = 12, p-value = 0.1573
```

```
archtest(adjusted_residuals, lag = 16)
```

```
##
##  Engle's LM ARCH Test
##
## data:  adjusted_residuals
## statistic = 68.164, lag = 16, p-value = 2.088e-08
## alternative hypothesis: ARCH effects of order 16 are present
```

The tests confirmed the presence of conditional heteroskedasticity among the residuals making the implementation of a volatility model quite useful to improve our estimate. We computed the adjusted and squared residuals and used the Ljung-box test again:

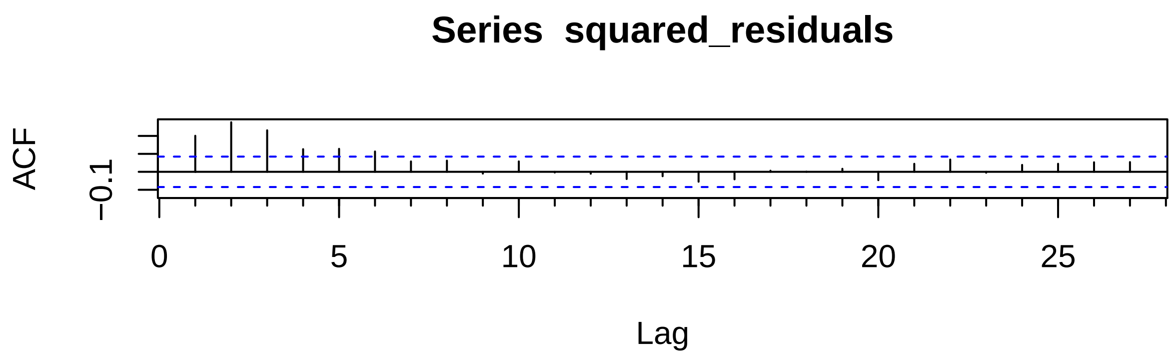
```
squared_residuals <- adjusted_residuals^2
Box.test(squared_residuals, lag = 12, type = "Ljung")
```

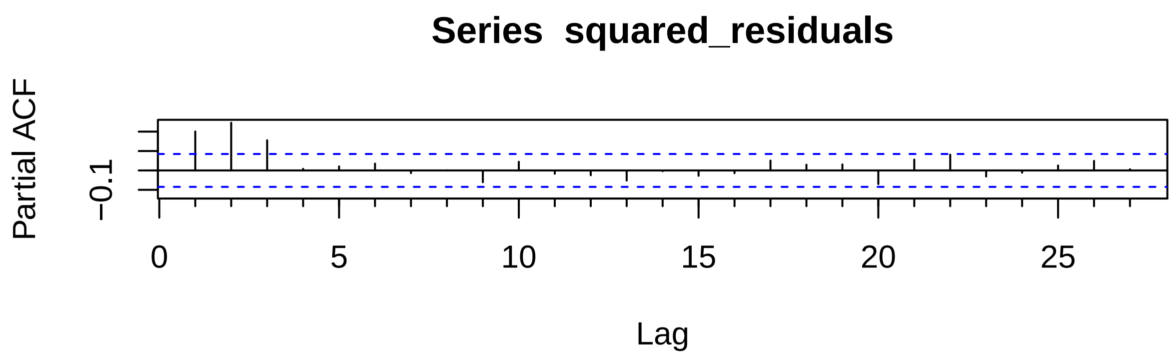
```
##
##  Box-Ljung test
##
## data:  squared_residuals
## X-squared = 122.22, df = 12, p-value < 2.2e-16
```

Confirming that there is still correlation.

We observed the ACF and PACF plot of the squared residuals in order to estimate an appropriate GARCH order:

```
par(mfrow=c(2,1))
Acf(squared_residuals)
Pacf(squared_residuals)
```






```
par(mfrow=c(1,1))
```

Model Estimation

The ACF plot showed sigificant spikes up to lag 6, while the PACF up to lag 4. We started with a simple GARCH(1,1) and then tried to increase and reduce the complexity looking at the significance of the coefficients:

```
mod1 <- garchFit(logbr ~ arma(1,0) + garch(1, 1), data = logbr,  trace = F,
                 include.mean = F)
summary(mod1)
```

```
##
## Title:
##  GARCH Modelling
##
```

```
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x000001db5278a0b0>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1
## 0.07194200  0.00020599  0.12865566  0.69263537
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.194e-02   4.735e-02    1.519 0.128700
## omega  2.060e-04   7.403e-05    2.783 0.005394 **
## alpha1 1.287e-01   3.875e-02    3.320 0.000899 ***
## beta1  6.926e-01   8.609e-02    8.046 8.88e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1064.384    normalized:  1.989502
##
## Description:
##  Tue May 20 20:32:10 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  27.3782166 1.134739e-06
##  Shapiro-Wilk Test  R    W       0.9862924 6.343748e-05
##  Ljung-Box Test     R    Q(10)  10.4429828 4.025251e-01
##  Ljung-Box Test     R    Q(15)  12.6293837 6.309000e-01
##  Ljung-Box Test     R    Q(20)  13.7406480 8.433984e-01
##  Ljung-Box Test     R^2  Q(10)  13.8768010 1.786856e-01
##  Ljung-Box Test     R^2  Q(15)  17.9999103 2.626703e-01
##  Ljung-Box Test     R^2  Q(20)  22.7611174 3.006759e-01
##  LM Arch Test       R    TR^2   15.1708863 2.322240e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.964051 -3.932034 -3.964162 -3.951524
```

```
mod2 <- garchFit(logbr ~ arma(1,0) + garch(2, 1), data = logbr,  trace = F,
                 include.mean = F)
summary(mod2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 1), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 1)
## <environment: 0x000001db4fe35d20>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2       beta1
## 0.07320112  0.00033345  0.05423235  0.13745800  0.51630578
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    0.0732011   0.0441788    1.657 0.097534 .
## omega  0.0003334   0.0001268    2.629 0.008557 **
## alpha1 0.0542323   0.0442447    1.226 0.220298
## alpha2 0.1374580   0.0593414    2.316 0.020537 *
## beta1  0.5163058   0.1507924    3.424 0.000617 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.013    normalized:  1.994416
##
## Description:
##  Tue May 20 20:32:10 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  19.2904042 6.473542e-05
##  Shapiro-Wilk Test  R    W       0.9886679 3.709832e-04
##  Ljung-Box Test     R    Q(10)   9.7233334 4.650923e-01
##  Ljung-Box Test     R    Q(15)  12.1372524 6.686118e-01
##  Ljung-Box Test     R    Q(20)  13.2998069 8.641532e-01
##  Ljung-Box Test     R^2  Q(10)   6.7860956 7.454727e-01
##  Ljung-Box Test     R^2  Q(15)   9.9585381 8.223398e-01
##  Ljung-Box Test     R^2  Q(20)  14.5562427 8.012117e-01
##  LM Arch Test       R    TR^2    6.8591981 8.667763e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
```

```
## -3.970141 -3.930120 -3.970314 -3.954483
```

```
mod3 <- garchFit(logbr ~ arma(1,0) + garch(2, 2), data = logbr,  trace = F,
                 include.mean = F)
summary(mod3)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 2), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 2)
## <environment: 0x000001db47970620>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2       beta1       beta2
## 0.07320001  0.00033345  0.05423569  0.13745458  0.51630670  0.00000001
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.320e-02   4.419e-02    1.656   0.0976 .
## omega  3.334e-04   1.275e-04    2.616   0.0089 **
## alpha1 5.424e-02   4.517e-02    1.201   0.2299
## alpha2 1.375e-01   5.942e-02    2.313   0.0207 *
## beta1  5.163e-01   2.715e-01    1.902   0.0572 .
## beta2  1.000e-08   2.252e-01    0.000   1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.013    normalized:  1.994416
##
## Description:
##  Tue May 20 20:32:11 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  19.2905775 6.472981e-05
##  Shapiro-Wilk Test  R    W       0.9886678 3.709611e-04
##  Ljung-Box Test     R    Q(10)   9.7233489 4.650909e-01
##  Ljung-Box Test     R    Q(15)  12.1372655 6.686108e-01
##  Ljung-Box Test     R    Q(20)  13.2998153 8.641528e-01
```

```
##  Ljung-Box Test     R^2  Q(10)   6.7861627 7.454665e-01
##  Ljung-Box Test     R^2  Q(15)   9.9585950 8.223363e-01
##  Ljung-Box Test     R^2  Q(20)  14.5563316 8.012068e-01
##  LM Arch Test       R    TR^2    6.8592561 8.667726e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.966403 -3.918377 -3.966651 -3.947613
```

```
mod4 <- garchFit(logbr ~ arma(1,0) + garch(2, 0), data = logbr,  trace = F,
                 include.mean = F)
summary(mod4)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(2, 0), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(2, 0)
## <environment: 0x000001db55f6a0e0>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2
## 0.07383483  0.00081678  0.10962761  0.18396544
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    7.383e-02   4.601e-02    1.605 0.108546
## omega  8.168e-04   7.699e-05   10.609  < 2e-16 ***
## alpha1 1.096e-01   4.396e-02    2.494 0.012632 *
## alpha2 1.840e-01   5.315e-02    3.461 0.000537 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1061.92    normalized:  1.984898
##
## Description:
##  Tue May 20 20:32:11 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
```

```
##  Jarque-Bera Test   R    Chi^2  15.3506087 0.0004641493
##  Shapiro-Wilk Test  R    W       0.9892767 0.0005967167
##  Ljung-Box Test     R    Q(10)  11.0778174 0.3514875454
##  Ljung-Box Test     R    Q(15)  14.3909757 0.4961070230
##  Ljung-Box Test     R    Q(20)  15.7282269 0.7333349925
##  Ljung-Box Test     R^2  Q(10)  19.3841648 0.0356459465
##  Ljung-Box Test     R^2  Q(15)  21.5786842 0.1193365252
##  Ljung-Box Test     R^2  Q(20)  29.2781574 0.0824250819
##  LM Arch Test       R    TR^2   17.4001565 0.1351545825
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.954843 -3.922826 -3.954954 -3.942316
```

```r
mod5 <- garchFit(logbr ~ arma(1,0) + garch(3, 0), data = logbr,  trace = F,
                 include.mean = F)
summary(mod5)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(3, 0), data = logbr,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(3, 0)
## <environment: 0x000001db5390b9d8>
##  [data = logbr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1       omega      alpha1      alpha2      alpha3
## 0.07862561  0.00072894  0.04896167  0.16188568  0.14784491
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     0.0786256   0.0430469    1.827 0.067774 .
## omega   0.0007289   0.0000761    9.579  < 2e-16 ***
## alpha1  0.0489617   0.0440887    1.111 0.266772
## alpha2  0.1618857   0.0479650    3.375 0.000738 ***
## alpha3  0.1478449   0.0527724    2.802 0.005086 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.923    normalized:  1.996118
##
```

```
## Description:
##  Tue May 20 20:32:11 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  16.8025891 0.0002245764
##  Shapiro-Wilk Test  R    W       0.9894586 0.0006890548
##  Ljung-Box Test     R    Q(10)   9.8129603 0.4570534704
##  Ljung-Box Test     R    Q(15)  12.0528878 0.6750215101
##  Ljung-Box Test     R    Q(20)  13.0296593 0.8761078779
##  Ljung-Box Test     R^2  Q(10)   6.6816631 0.7551171747
##  Ljung-Box Test     R^2  Q(15)   9.3677800 0.8575158661
##  Ljung-Box Test     R^2  Q(20)  13.9219512 0.8344310329
##  LM Arch Test       R    TR^2    6.4462092 0.8919413738
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.973544 -3.933523 -3.973717 -3.957886
```

```
get_model_stats <- function(model) {
  ll <- model@fit$llh
  aic <- model@fit$ics["AIC"]
  bic <- model@fit$ics["BIC"]
  return(c(LogLikelihood = ll, AIC = aic, BIC = bic))
}
model_summary <- data.frame(
  Model = paste0("mod", 1:5),
  t(sapply(list(mod1, mod2, mod3, mod4, mod5), get_model_stats))
)
print(model_summary)
```

```
##   Model LogLikelihood.LogLikelihood   AIC.AIC   BIC.BIC
## 1  mod1                   -1064.384 -3.964051 -3.932034
## 2  mod2                   -1067.013 -3.970141 -3.930120
## 3  mod3                   -1067.013 -3.966403 -3.918377
## 4  mod4                   -1061.920 -3.954843 -3.922826
## 5  mod5                   -1067.923 -3.973544 -3.933523
```

To improve furthermore the model different conditional distributions were tested:

```
mod1st <- garchFit(logbr ~ arma(1,0) + garch(1, 1), data = logbr,  trace = F,
                   include.mean = F, cond.dist = "std")
summary(mod1st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     cond.dist = "std", include.mean = F, trace = F)
```

```
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x000001db50339c08>
##  [data = logbr]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1       shape
## 6.7174e-02  1.8258e-04  1.3416e-01  7.1182e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    6.717e-02   4.594e-02    1.462  0.14372
## omega  1.826e-04   7.679e-05    2.378  0.01742 *
## alpha1 1.342e-01   4.451e-02    3.014  0.00258 **
## beta1  7.118e-01   9.055e-02    7.861 3.77e-15 ***
## shape  1.000e+01   4.178e+00    2.394  0.01668 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1067.739    normalized:  1.995773
##
## Description:
##  Tue May 20 20:32:11 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  28.8032884 5.564747e-07
##  Shapiro-Wilk Test  R    W       0.9860708 5.416921e-05
##  Ljung-Box Test     R    Q(10)  10.3390443 4.112681e-01
##  Ljung-Box Test     R    Q(15)  12.4686298 6.432669e-01
##  Ljung-Box Test     R    Q(20)  13.5537372 8.523830e-01
##  Ljung-Box Test     R^2  Q(10)  12.7437277 2.383553e-01
##  Ljung-Box Test     R^2  Q(15)  17.2880259 3.019398e-01
##  Ljung-Box Test     R^2  Q(20)  21.9752284 3.418563e-01
##  LM Arch Test       R    TR^2   14.2580455 2.845344e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.972855 -3.932834 -3.973027 -3.957196
```

```r
mod1sst <- garchFit(logbr ~ arma(1,0) + garch(1, 1), data = logbr,  trace = F,
                    include.mean = F, cond.dist = "sstd")
summary(mod1sst)
```

```
##
```

```
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = logbr ~ arma(1, 0) + garch(1, 1), data = logbr,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x000001db56423d90>
##  [data = logbr]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##        ar1       omega      alpha1       beta1        skew       shape
## 5.0426e-02  1.7664e-04  1.3293e-01  7.2257e-01  8.5842e-01  1.0000e+01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    5.043e-02   4.525e-02    1.114  0.26511
## omega  1.766e-04   7.270e-05    2.430  0.01511 *
## alpha1 1.329e-01   4.265e-02    3.117  0.00183 **
## beta1  7.226e-01   8.469e-02    8.531  < 2e-16 ***
## skew   8.584e-01   5.000e-02   17.168  < 2e-16 ***
## shape  1.000e+01   4.230e+00    2.364  0.01809 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1071.33    normalized:  2.002486
##
## Description:
##  Tue May 20 20:32:11 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  29.4833585 3.960683e-07
##  Shapiro-Wilk Test  R    W       0.9859208 4.870702e-05
##  Ljung-Box Test     R    Q(10)  10.6576958 3.848008e-01
##  Ljung-Box Test     R    Q(15)  12.7490505 6.216742e-01
##  Ljung-Box Test     R    Q(20)  13.8783098 8.366119e-01
##  Ljung-Box Test     R^2  Q(10)  12.3672716 2.612280e-01
##  Ljung-Box Test     R^2  Q(15)  16.9685614 3.207473e-01
##  Ljung-Box Test     R^2  Q(20)  21.5071926 3.678387e-01
##  LM Arch Test       R    TR^2   14.0699732 2.962623e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
```

```
## -3.982541 -3.934516 -3.982789 -3.963751
```

In both cases the standard residuals are still not normally distributed but not serially correlated. The shape coefficient is significant in both models as the skew coefficient in the second one.

Again we defined a dataframe to compare the models statistics:

```
model_summary <- data.frame(
  Model = c("mod1st", "mod1sst"),
  t(sapply(list(mod1st, mod1sst), get_model_stats))
)
print(model_summary)
```

```
##     Model LogLikelihood.LogLikelihood   AIC.AIC   BIC.BIC
## 1  mod1st                   -1067.739 -3.972855 -3.932834
## 2 mod1sst                   -1071.330 -3.982541 -3.934516
```

Our best model is still an ARMA(1,0) + GARCH(1,1) - SSTD

Fourth Forecast

We then used this model to compute our forecast:

```
f = predict(mod1sst, n.ahead = 1)
f
```

```
##    meanForecast  meanError standardDeviation
## 1 -0.0003344157 0.03402372        0.03402372
```

```
f$meanForecast
```

```
## [1] -0.0003344157
```

```
last <- tail(data$BLACKROCK, 1)
log_last <- log(last)
log_forecast <- log_last + f$meanForecast
forecast <- exp(log_forecast)
forecast
```

```
## [1] 961.5185
```

Out week4 forecast is: 961.5184.

Fourth Error

The actual value registered for out stock was: 961.8401, we compute the errors:

```
real_price <- 961.8401
abs_error = abs(real_price - forecast)
perc_error = (abs_error / real_price) * 100

cat("Absolute error :", abs_error, "\n")
```

```
## Absolute error : 0.3216007
```

```
cat("Percentage error :", perc_error, "%\n\n")
```

```
## Percentage error : 0.03343598 %
```

WEEK 5

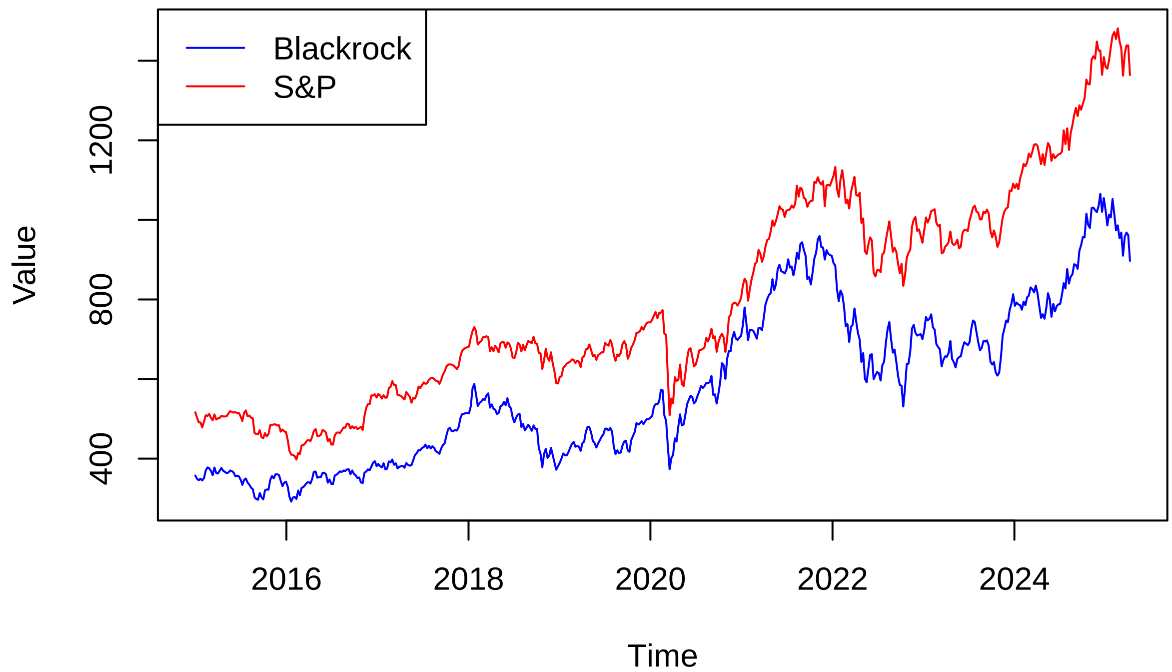
```
data = read_excel(here("blackrock.xlsx"), sheet = 5, col_names = TRUE)[c(1,2,3)]
```

```
## New names:
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...8`
## * `` -> `...9`
```

Introduction

Now we compare two approaches to forecasting the weekly stock price of BlackRock using ARMA-GARCH models with an exogenous variable: the S&P500 FIN SVS index. The fluctuations of the S&P 500 influence the value of BlackRock's assets, as many of its investments are tied to the stocks that make up the index, directly impacting the price of its shares. Including the S&P 500 as an exogenous variable could help improve the predictive power of the model, as it introduces potentially relevant external information. In particular, the index may capture co-movements between BlackRock's stock and the broader U.S. equity market.

```
br = data$BLACKROCK
sp = data$`S&P 500 FIN SVS - PRICE INDEX`

ylim_range <- range(c(data[[2]], data[[3]]), na.rm = TRUE)
plot(data[[1]], data[[2]], type = "l", col = "blue", xlab = "Time",
     ylab = "Value", ylim = ylim_range)
lines(data[[1]], data[[3]], col = "red")
legend("topleft", legend = c("Blackrock", "S&P"), col = c("blue", "red"),
       lty = 1)
```



- Approach 1 uses the lagged S&P500, i.e., the value of the index at time t to forecast BlackRock at time t+1.
- Approach 2 first predicts the S&P500 at time t+1 using a dedicated model, and then uses this forecast as an exogenous input to estimate BlackRock at the same horizon.

First, we split the data into a training set (90%) and a test set (10%).

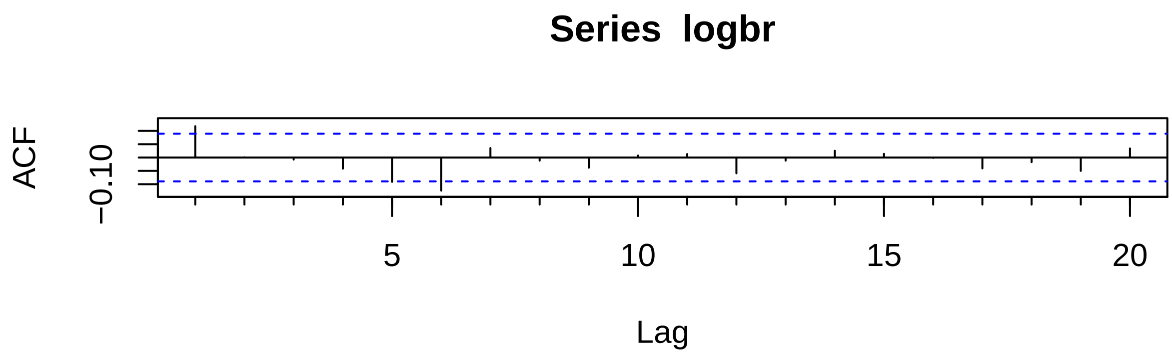
```
train_size <- floor(0.9 * nrow(data))
train <- data[1:train_size, ]
test <- data[(train_size+1):nrow(data), ]
```

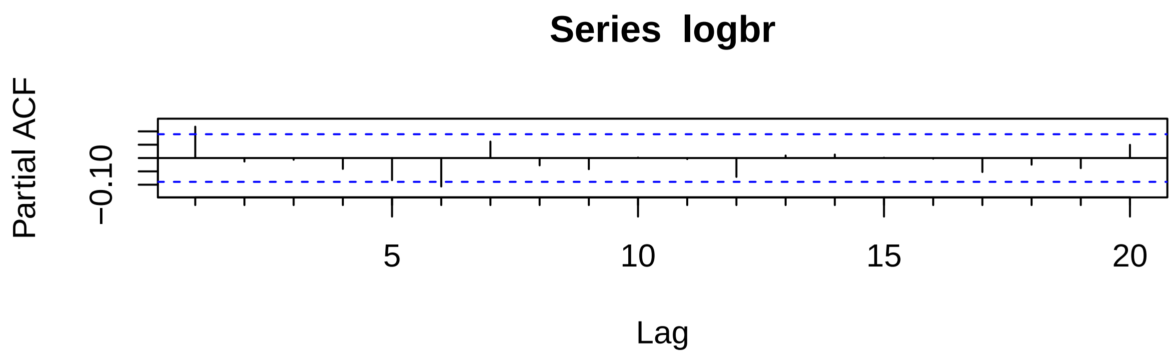
```
br = train$BLACKROCK
logbr = diff(log(br))

t.test(logbr)
```

```
##
##  One Sample t-test
##
## data:  logbr
## t = 1.1082, df = 481, p-value = 0.2683
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.001360597  0.004880398
```

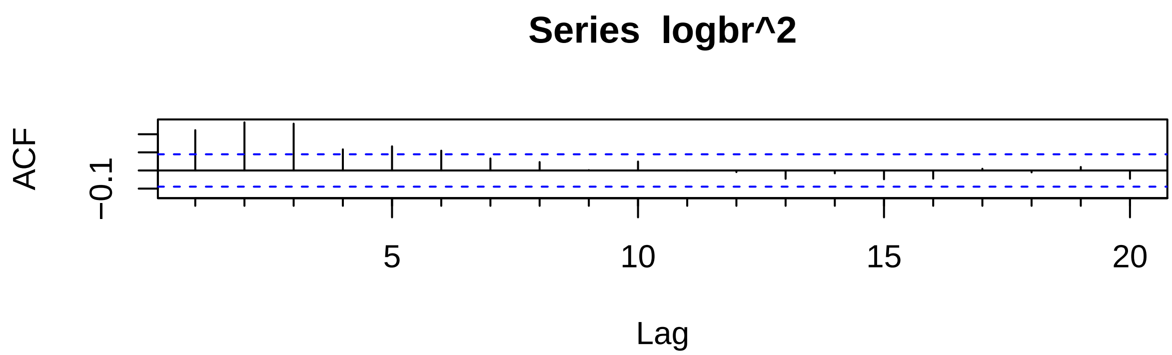
```
## sample estimates:
##   mean of x
## 0.001759901
```

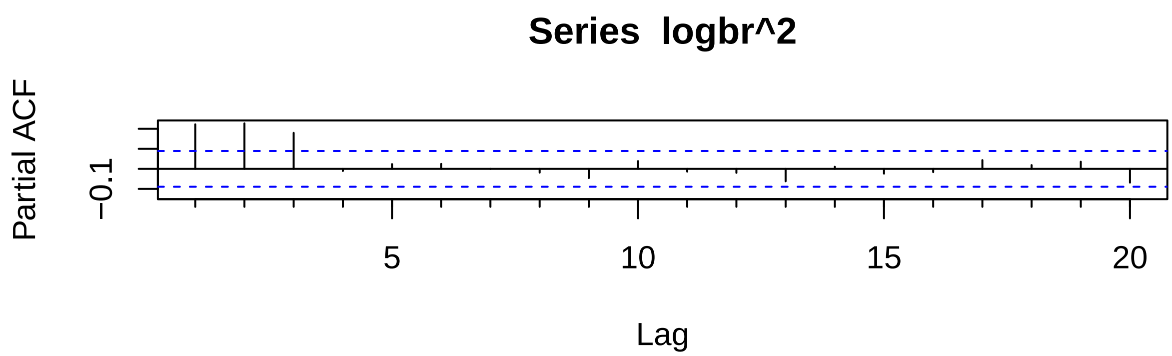
```r
par(mfrow=c(2,1))
Acf(logbr, lag.max = 20)
Pacf(logbr, lag.max=20)
```






```r
par(mfrow=c(1,1))

par(mfrow=c(2,1))
Acf(logbr^2, lag.max = 20)
Pacf(logbr^2, lag.max=20)
```






```
par(mfrow=c(1,1))
```

The t-test on log returns indicates that the mean is not significantly different from zero, suggesting that a constant term may not be necessary in the ARMA model.

The ACF and PACF plots of the log returns show significant autocorrelations at various lags, indicating potential ARMA structures. Candidate models include ARMA(1,1), ARMA(6,6), ARMA(6,1), and ARMA(1,6).

The ACF and PACF of the squared returns suggest the presence of volatility, indicating that a GARCH component may be needed.

Fist approach

After lagging the S&P500 index, we verify that the lengths of the two series are equal. This ensures that the series are temporally shifted by one period, which is necessary for modeling BlackRock returns with the lagged S&P500 index.

```
train <- train %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
train <- na.omit(train)
sp_lag <- train$SP_LAG[-1]
logbr <- diff(log(train$BLACKROCK))[-length(train$BLACKROCK)]
print(length(logbr) == length(sp_lag))
```

```
## [1] TRUE
```

The idea is comparing ARMAX-GARCH and ARMA-GARCHX models. Both models include the lagged S&P500 index as an exogenous variable, and we aim to determine which model fits best by evaluating their performance in capturing both time-series dependencies and volatility dynamics.

First phase of first approach

Several ARMAX specifications are estimated in order to identify the optimal structure of the autoregressive and moving average components.

```
armax11 = Arima(logbr, order=c(1,0,1), xreg=sp_lag, include.mean = F)
summary(armax11)
```

```
## Series: logbr
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1     ma1  xreg
##       0.0195  0.0995     0
## s.e.  0.3974  0.3928     0
##
## sigma^2 = 0.001207:  log likelihood = 935.11
## AIC=-1862.23   AICc=-1862.14   BIC=-1845.52
##
## Training set error measures:
##                        ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.0007087515 0.03462928 0.02568187 127.2984 150.1772 0.7261806
##                      ACF1
## Training set 0.0002395769
```

```
confint(armax11)
```

```
##              2.5 %       97.5 %
## ar1  -0.7593392820 7.983107e-01
## ma1  -0.6704136701 8.693643e-01
## xreg -0.0000931607 9.593045e-05
```

```
armax66 = Arima(logbr, order=c(6,0,6), xreg=sp_lag, include.mean = F)
summary(armax66)
```

```
## Series: logbr
## Regression with ARIMA(6,0,6) errors
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): Si è prodotto un NaN
```

```
##          ar1      ar2     ar3      ar4     ar5     ar6      ma1     ma2
##       0.2108  -0.3397  0.1202  -0.2674  0.1347  0.4909  -0.1174  0.3026
## s.e.  0.1860   0.2468  0.2150   0.2415  0.1800  0.2100   0.1171  0.1968
##          ma3     ma4      ma5      ma6  xreg
##      -0.0817  0.2519  -0.1529  -0.6409     0
```

```
## s.e.   0.1286  0.1940   0.0867   0.1893   NaN
##
## sigma^2 = 0.001172:  log likelihood = 946.12
## AIC=-1864.23   AICc=-1863.33   BIC=-1805.77
##
## Training set error measures:
##                        ME      RMSE        MAE      MPE     MAPE      MASE
## Training set 0.0006568475 0.0337715 0.02497044 150.6243 206.0361 0.7060643
##                    ACF1
## Training set 0.01802178
```

```r
confint(armax66)
```

```
## Warning in sqrt(diag(vcov(object))): Si è prodotto un NaN
```

```
##            2.5 %      97.5 %
## ar1  -0.15380230  0.57537268
## ar2  -0.82350004  0.14410020
## ar3  -0.30117281  0.54152311
## ar4  -0.74072494  0.20594409
## ar5  -0.21799722  0.48745084
## ar6   0.07932285  0.90237811
## ma1  -0.34691496  0.11221143
## ma2  -0.08324860  0.68837620
## ma3  -0.33377732  0.17032741
## ma4  -0.12828301  0.63214012
## ma5  -0.32287578  0.01701119
## ma6  -1.01198185 -0.26989162
## xreg         NaN         NaN
```

```r
armax16 = Arima(logbr, order=c(1,0,6), xreg=sp_lag, include.mean = F)
summary(armax16)
```

```
## Series: logbr
## Regression with ARIMA(1,0,6) errors
##
## Coefficients:
##           ar1     ma1     ma2      ma3      ma4      ma5      ma6   xreg
##       -0.1507  0.2650  0.0180  -0.0163  -0.0376  -0.0738  -0.1546  0e+00
## s.e.   0.2854  0.2873  0.0657   0.0544   0.0537   0.0511   0.0492  1e-04
##
## sigma^2 = 0.001189:  log likelihood = 941.03
## AIC=-1864.06   AICc=-1863.68   BIC=-1826.48
##
## Training set error measures:
##                        ME       RMSE       MAE      MPE     MAPE      MASE
## Training set 0.0006943462 0.03420046 0.0257322 150.8594 188.5013 0.7276037
##                       ACF1
## Training set -0.0005647362
```

```r
confint(armax16)
```

```
##              2.5 %        97.5 %
## ar1  -0.7101797460  0.4086871821
## ma1  -0.2980329500  0.8279750749
## ma2  -0.1107068990  0.1467502398
## ma3  -0.1230538065  0.0903839210
## ma4  -0.1428468712  0.0676212576
## ma5  -0.1740267469  0.0263556638
## ma6  -0.2510910254 -0.0581905864
## xreg -0.0001299823  0.0001332044
```

```r
armax61 = Arima(logbr, order=c(6,0,1), xreg=sp_lag, include.mean = F)
summary(armax61)
```

```
## Series: logbr
## Regression with ARIMA(6,0,1) errors
##
## Coefficients:
##           ar1     ar2      ar3      ar4      ar5      ar6     ma1   xreg
##       -0.3001  0.0322  -0.0078  -0.0318  -0.0796  -0.1428  0.4131  0e+00
## s.e.   0.2679  0.0592   0.0490   0.0488   0.0492   0.0463  0.2705  1e-04
##
## sigma^2 = 0.001192:  log likelihood = 940.59
## AIC=-1863.18   AICc=-1862.8   BIC=-1825.6
##
## Training set error measures:
##                        ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.0007023684 0.03423249 0.02571962 153.6629 186.6335 0.7272481
##                     ACF1
## Training set 0.001353267
```

```r
confint(armax61)
```

```
##              2.5 %        97.5 %
## ar1  -0.8251172012  0.2248659672
## ar2  -0.0838042383  0.1481355946
## ar3  -0.1037375418  0.0881802170
## ar4  -0.1274201484  0.0637276770
## ar5  -0.1759130384  0.0167960807
## ar6  -0.2335538267 -0.0521124828
## ma1  -0.1169974640  0.9432174416
## xreg -0.0001010146  0.0001041323
```

```r
ar1 = Arima(logbr, order=c(1,0,0), xreg=sp_lag, include.mean = F)
summary(ar1)
```

```
## Series: logbr
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
```

```
##          ar1  xreg
##       0.1179     0
## s.e.  0.0470     0
##
## sigma^2 = 0.001204:  log likelihood = 935.08
## AIC=-1864.16   AICc=-1864.11   BIC=-1851.64
##
## Training set error measures:
##                        ME       RMSE        MAE     MPE    MAPE      MASE
## Training set 0.0007173495 0.03463157 0.02569291 125.321 148.427 0.7264927
##                     ACF1
## Training set 0.001301107
```

```
confint(ar1)
```

```
##               2.5 %       97.5 %
## ar1   2.586559e-02 2.099959e-01
## xreg -9.154827e-05 9.426859e-05
```

```
cbind(AIC(ar1), AIC(armax61), AIC(armax16), AIC(armax66), AIC(armax11))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1864.163 -1863.183 -1864.063 -1864.233 -1862.227
```

```
cbind(BIC(ar1), BIC(armax61), BIC(armax16), BIC(armax66), BIC(armax11))
```

```
##           [,1]    [,2]      [,3]      [,4]      [,5]
## [1,] -1851.636 -1825.6 -1826.481 -1805.771 -1845.523
```

Model selection is supported by comparing AIC and BIC values across specifications. Although the AIC values are relatively close, the BIC clearly favors the more parsimonious AR(1) model, suggesting that adding further lags does not significantly improve model fit.

This section forecasts with different ARIMAX models using the test set and calculates the RMSE to evaluate the prediction accuracy of each model.

```
log_diff_test_br <- diff(log(test$BLACKROCK))[-length(test$BLACKROCK)]
test <- test %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
test <- na.omit(test) #toglie la prima riha NA

predictions11 <- forecast(armax11, xreg=test$SP_LAG, h=53)
predictions66 <- forecast(armax66, xreg=test$SP_LAG, h=53)
predictions16 <- forecast(armax16, xreg=test$SP_LAG, h=53)
predictions61 <- forecast(armax61, xreg=test$SP_LAG, h=53)
predictions1 <- forecast(ar1, xreg=test$SP_LAG, h=53)

rmse11 <- sqrt(mean((predictions11$mean - log_diff_test_br)^2))
rmse66 <- sqrt(mean((predictions66$mean - log_diff_test_br)^2))
rmse16 <- sqrt(mean((predictions16$mean - log_diff_test_br)^2))
rmse61 <- sqrt(mean((predictions61$mean - log_diff_test_br)^2))
rmse1 <- sqrt(mean((predictions1$mean - log_diff_test_br)^2))

print(paste("RMSE for ARIMAX(1,0,1):", rmse11))
```

```
## [1] "RMSE for ARIMAX(1,0,1): 0.032909449491259"
```

```r
print(paste("RMSE for ARIMAX(6,0,6):", rmse66))
```

```
## [1] "RMSE for ARIMAX(6,0,6): 0.033240579745032"
```

```r
print(paste("RMSE for ARIMAX(1,0,6):", rmse16))
```

```
## [1] "RMSE for ARIMAX(1,0,6): 0.0329011584676921"
```

```r
print(paste("RMSE for ARIMAX(6,0,1):", rmse61))
```

```
## [1] "RMSE for ARIMAX(6,0,1): 0.0328840614383612"
```

```r
print(paste("RMSE for ARIMAX(1,0,0):", rmse1))
```

```
## [1] "RMSE for ARIMAX(1,0,0): 0.0329102968339991"
```

Although BIC favors the simpler AR(1) model, RMSE values—being directly linked to out-of-sample predictive accuracy—suggest a marginal advantage for the ARMAX(6,1) specification. Given the forecasting objective, the ARMAX(6,1) model appears to offer the best trade-off between complexity and predictive performance, despite its higher BIC.

To assess the adequacy of the ARMAX(6,1) model, residual diagnostics are performed.

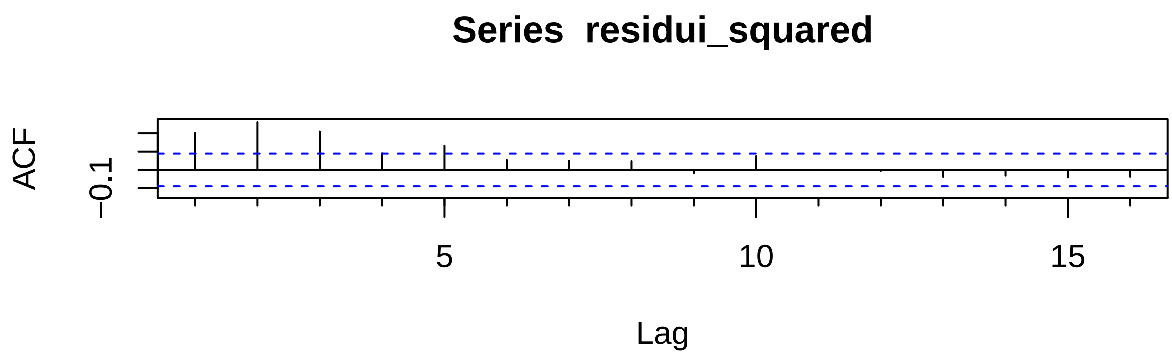
```r
residuals_arimax <- armax61$residuals
Box.test(residuals_arimax, lag = 12, type = "Ljung")
```

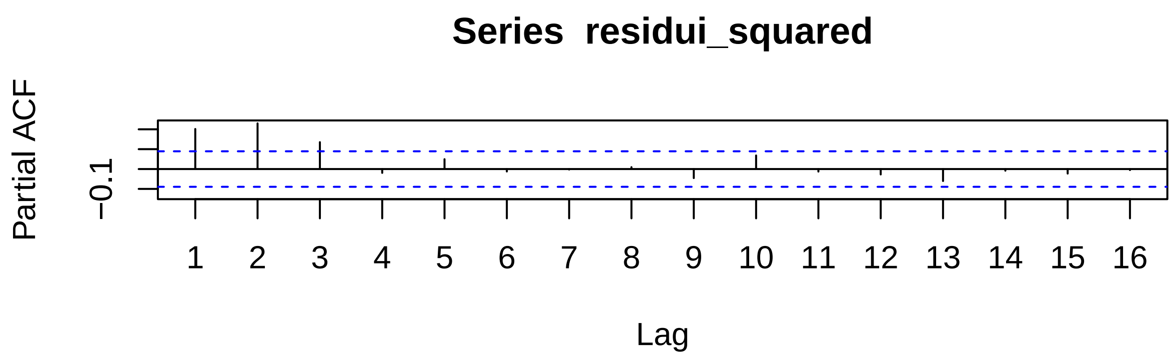
```
##
##  Box-Ljung test
##
## data:  residuals_arimax
## X-squared = 4.1138, df = 12, p-value = 0.9813
```

```r
residui_squared <- residuals_arimax^2
Box.test(residui_squared, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residui_squared
## X-squared = 93.483, df = 12, p-value = 1.044e-14
```

```r
par(mfrow=c(2,1))
Acf(residui_squared, lag.max = 16)
Pacf(residui_squared, lag.max = 16)
```






```r
par(mfrow=c(1,1))
```

The Box-Ljung test on the residuals of the ARIMAX(6,0,1) model shows a high p-value (> 0.05), indicating that there is no significant autocorrelation in the residuals, suggesting that the model is well-specified. However, the Box-Ljung test on the squared residuals indicates significant autocorrelation in the squared residuals. This suggests the presence of heteroscedasticity, implying that a GARCH model may be needed to capture the volatility. The ACF and PACF on the squared residuals suggest that GARCH(1,1) or ARCH(3,0) models may be appropriate.

```r
g61x11 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "norm"
)

garch61x11 <- ugarchfit(spec = g61x11, data = logbr)

g61x30 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0)),
  mean.model = list(armaOrder = c(6,1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "norm"
)

garch61x30 <- ugarchfit(spec = g61x30, data = logbr)
```

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

```r
# fail to converge

g61x21 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(6,1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "norm"
)

garch61x21 <- ugarchfit(spec = g61x21, data = logbr)

predictions_g61x11 <- ugarchforecast(garch61x11, n.ahead = 53,
                                     external.forecasts =
                                       list(mregfor =test$SP_LAG))
predictions_g61x21 <- ugarchforecast(garch61x21, n.ahead = 53,
                                     external.forecasts =
                                       list(mregfor =test$SP_LAG))

rmse_g61x11 <- sqrt(mean((predictions_g61x11@forecast$seriesFor
                          - log_diff_test_br)^2))
rmse_g61x21 <- sqrt(mean((predictions_g61x21@forecast$seriesFor
                          - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(6,1)-GARCH(1,1):", rmse_g61x11))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1): 0.0329309808862437"
```

```r
print(paste("RMSE for ARMAX(6,1)-GARCH(2,1):", rmse_g61x21))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(2,1): 0.0329915350054456"
```

```r
infocriteria(garch61x11)*length(train$SP_LAG)
```

```
##
## Akaike       -1901.815
## Bayes        -1855.785
## Shibata      -1902.305
## Hannan-Quinn -1883.724
```

```r
infocriteria(garch61x21)*length(train$SP_LAG)
```

```
##
## Akaike       -1905.623
## Bayes        -1855.409
## Shibata      -1906.204
## Hannan-Quinn -1885.887
```

Between the two GARCH specifications, the ARMAX(6,1)-GARCH(1,1) model yields a slightly lower RMSE, indicating better forecasting accuracy. Although the ARMAX(6,1)-GARCH(2,1) model presents slightly better AIC and Shibata criteria, its higher RMSE makes it less attractive from a predictive standpoint.

```
garch61x11
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.217781    0.272267 -0.79988 0.423780
## ar2     0.021257    0.055203  0.38506 0.700190
## ar3    -0.018825    0.047984 -0.39232 0.694823
## ar4    -0.029308    0.047513 -0.61685 0.537334
## ar5    -0.052737    0.046253 -1.14018 0.254211
## ar6    -0.111038    0.046091 -2.40911 0.015992
## ma1     0.315488    0.270557  1.16607 0.243588
## mxreg1  0.000002    0.000002  1.33267 0.182641
## omega   0.000198    0.000073  2.71029 0.006722
## alpha1  0.140391    0.044453  3.15817 0.001588
## beta1   0.688313    0.087827  7.83715 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.217781    0.173698 -1.25379 0.209919
## ar2     0.021257    0.044038  0.48269 0.629314
## ar3    -0.018825    0.045168 -0.41678 0.676840
## ar4    -0.029308    0.040604 -0.72181 0.470410
## ar5    -0.052737    0.042683 -1.23555 0.216626
## ar6    -0.111038    0.044656 -2.48649 0.012901
## ma1     0.315488    0.174360  1.80940 0.070389
## mxreg1  0.000002    0.000002  1.22161 0.221854
## omega   0.000198    0.000067  2.95932 0.003083
## alpha1  0.140391    0.045549  3.08218 0.002055
## beta1   0.688313    0.080066  8.59677 0.000000
##
## LogLikelihood : 959.9349
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9457
## Bayes        -3.8502
## Shibata      -3.9467
```

```
## Hannan-Quinn -3.9081
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2233  0.6366
## Lag[2*(p+q)+(p+q)-1][20]    2.3633  1.0000
## Lag[4*(p+q)+(p+q)-1][34]    9.8917  0.9979
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                        statistic p-value
## Lag[1]                     1.492 0.22190
## Lag[2*(p+q)+(p+q)-1][5]     7.433 0.04068
## Lag[4*(p+q)+(p+q)-1][9]     8.895 0.08541
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.9412 0.500 2.000  0.3320
## ARCH Lag[5]    1.9859 1.440 1.667  0.4744
## ARCH Lag[7]    2.1785 2.315 1.543  0.6795
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.125
## Individual Statistics:
## ar1    0.14187
## ar2    0.07725
## ar3    0.08833
## ar4    0.11122
## ar5    0.04210
## ar6    0.09395
## ma1    0.12883
## mxreg1 0.07779
## omega  0.17135
## alpha1 0.21075
## beta1  0.18191
##
## 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           1.2720 0.2040
## Negative Sign Bias  1.0506 0.2940
## Positive Sign Bias  0.9699 0.3326
## Joint Effect        5.7650 0.1236
##
```

```
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     21.91      0.28873
## 2    30     35.36      0.19291
## 3    40     45.49      0.22010
## 4    50     65.47      0.05793
##
##
## Elapsed time : 0.353755
```

```
garch61x21
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.287878    0.312175 -0.92217 0.356441
## ar2     0.021959    0.060781  0.36128 0.717888
## ar3    -0.027951    0.050026 -0.55873 0.576349
## ar4    -0.026039    0.048204 -0.54018 0.589071
## ar5    -0.052809    0.045592 -1.15831 0.246738
## ar6    -0.098154    0.045876 -2.13955 0.032391
## ma1     0.393494    0.311574  1.26292 0.206617
## mxreg1  0.000003    0.000002  1.61332 0.106675
## omega   0.000316    0.000119  2.65201 0.008001
## alpha1  0.052646    0.048859  1.07750 0.281258
## alpha2  0.156205    0.065507  2.38455 0.017100
## beta1   0.514340    0.145887  3.52560 0.000423
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.287878    0.226697 -1.26988 0.204129
## ar2     0.021959    0.048593  0.45190 0.651342
## ar3    -0.027951    0.044959 -0.62170 0.534141
## ar4    -0.026039    0.039525 -0.65881 0.510020
## ar5    -0.052809    0.043791 -1.20592 0.227846
## ar6    -0.098154    0.042932 -2.28626 0.022239
## ma1     0.393494    0.223141  1.76344 0.077827
## mxreg1  0.000003    0.000002  1.49109 0.135937
## omega   0.000316    0.000125  2.53540 0.011232
## alpha1  0.052646    0.046539  1.13123 0.257957
## alpha2  0.156205    0.075899  2.05808 0.039583
```

```
## beta1   0.514340    0.157389  3.26796 0.001083
##
## LogLikelihood : 962.8349
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9536
## Bayes        -3.8494
## Shibata      -3.9548
## Hannan-Quinn -3.9126
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1328  0.7155
## Lag[2*(p+q)+(p+q)-1][20]    2.2843  1.0000
## Lag[4*(p+q)+(p+q)-1][34]   10.0522  0.9974
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1171  0.7322
## Lag[2*(p+q)+(p+q)-1][8]     2.2638  0.8164
## Lag[4*(p+q)+(p+q)-1][14]    5.0609  0.7619
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]     1.173 0.500 2.000  0.2787
## ARCH Lag[6]     1.910 1.461 1.711  0.5110
## ARCH Lag[8]     2.167 2.368 1.583  0.7077
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.3059
## Individual Statistics:
## ar1    0.11588
## ar2    0.09673
## ar3    0.08573
## ar4    0.11167
## ar5    0.06493
## ar6    0.11529
## ma1    0.11562
## mxreg1 0.05896
## omega  0.16897
## alpha1 0.20181
## alpha2 0.23863
## beta1  0.21565
##
## 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           0.5623 0.5742
## Negative Sign Bias  0.2608 0.7944
## Positive Sign Bias  0.6308 0.5285
## Joint Effect        2.9804 0.3947
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     18.92      0.46218
## 2    30     30.62      0.38348
## 3    40     40.66      0.39704
## 4    50     67.34      0.04209
##
##
## Elapsed time : 0.3459489
```

Although the ARMAX(6,1)-GARCH(1,1) model shows a slightly lower RMSE, it faces issues with the standardized squared residuals, suggesting potential misspecification. On the other hand, the ARMAX(6,1)-GARCH(2,1) model does not exhibit such problems, and both AIC and BIC values indicate a better fit. Given the marginal RMSE difference and the residual behavior, the ARMAX(6,1)-GARCH(2,1) model is preferred for its superior stability and fit.

Moreover, many ARMA components of the selected model are not significant, suggesting that simplifying the model could enhance efficiency without compromising predictive performance.

```
g11x21 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1,1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "norm"
)

garch11x21 <- ugarchfit(spec = g11x21, data = logbr)

predictions_g11x21 <- ugarchforecast(garch11x21, n.ahead = 53,
                  external.forecasts =list(mregfor=test$SP_LAG))
rmse_g11x21 <- sqrt(mean((predictions_g11x21@forecast$seriesFor
                         -log_diff_test_br)^2))

print(paste("RMSE for ARMAX(6,1)-GARCH(1,1):", rmse_g61x21))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1): 0.0329915350054456"
```

```
print(paste("RMSE for ARMAX(1,1)-GARCH(1,1):", rmse_g11x21))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(1,1): 0.0329780922677813"
```

```r
infocriteria(garch61x21)*length(train$SP_LAG)
```

```
##
## Akaike       -1905.623
## Bayes        -1855.409
## Shibata      -1906.204
## Hannan-Quinn -1885.887
```

```r
infocriteria(garch11x21)*length(train$SP_LAG)
```

```
##
## Akaike       -1910.876
## Bayes        -1881.584
## Shibata      -1911.076
## Hannan-Quinn -1899.363
```

The simplified ARMAX(1,1)-GARCH(2,1) model achieves a slightly lower RMSE, and both its AIC and BIC are smaller compared to the ARMAX(6,1)-GARCH(2,1). Given these improvements in both predictive performance and fit, the simplified model is preferred.

The performance of the model is evaluated using different distributional assumptions for the error terms.

```r
g11x21std <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "std"
)

garch11x21std <- ugarchfit(spec = g11x21std, data = logbr)

predictions_g11x21std<- ugarchforecast(garch11x21std, n.ahead = 53,
        external.forecasts = list(mregfor =test$SP_LAG))
rmse_g11x21std <- sqrt(mean((predictions_g11x21std@forecast$seriesFor
                             - log_diff_test_br)^2))

g11x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd"
)

garch11x21sstd <- ugarchfit(spec = g11x21sstd, data =logbr)

predictions_g11x21sstd <- ugarchforecast(garch11x21sstd, n.ahead = 53,
                        external.forecasts = list(mregfor =test$SP_LAG))
rmse_g11x21sstd <- sqrt(mean((predictions_g11x21sstd@forecast$seriesFor - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(1,1)-GARCH(2,1):", rmse_g11x21))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(2,1): 0.0329780922677813"
```

```r
print(paste("RMSE for ARMAX(1,1)-GARCH(2,1)-STD:", rmse_g11x21std))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(2,1)-STD: 0.0331658794612077"
```

```r
print(paste("RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD:", rmse_g11x21sstd))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD: 0.0329888512549517"
```

```r
infocriteria(garch11x21)*length(train$SP_LAG)
```

```
##
## Akaike       -1910.876
## Bayes        -1881.584
## Shibata      -1911.076
## Hannan-Quinn -1899.363
```

```r
infocriteria(garch11x21std)*length(train$SP_LAG)
```

```
##
## Akaike       -1919.185
## Bayes        -1885.709
## Shibata      -1919.446
## Hannan-Quinn -1906.028
```

```r
infocriteria(garch11x21sstd)*length(train$SP_LAG)
```

```
##
## Akaike       -1921.990
## Bayes        -1884.329
## Shibata      -1922.320
## Hannan-Quinn -1907.188
```

The ARMAX(1,1)-GARCH(2,1) model provides the best predictive performance with the lowest RMSE. However, given the negligible difference in RMSE between the normal and SSTD distributions, the latter is preferred. This choice is supported by its better fit, as indicated by the lower AIC and BIC values, which suggest that the SSTD distribution offers a more optimal balance between fit and predictive accuracy. The chosen model is ARMAX(1,1)-GARCH(2,1)-SSTD.

```r
garch11x21sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(1,0,1)
```

```
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.232425    0.362572 -0.64105 0.521493
## ma1     0.332289    0.353346  0.94041 0.347009
## mxreg1  0.000003    0.000002  1.43457 0.151409
## omega   0.000237    0.000100  2.35781 0.018383
## alpha1  0.043136    0.053928  0.79990 0.423772
## alpha2  0.152438    0.070223  2.17076 0.029949
## beta1   0.603635    0.124197  4.86030 0.000001
## skew    0.871019    0.054743 15.91093 0.000000
## shape   8.688614    3.715518  2.33847 0.019363
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.232425    0.295783 -0.78580 0.431987
## ma1     0.332289    0.294684  1.12761 0.259484
## mxreg1  0.000003    0.000002  1.36515 0.172207
## omega   0.000237    0.000081  2.94012 0.003281
## alpha1  0.043136    0.051782  0.83303 0.404827
## alpha2  0.152438    0.068201  2.23513 0.025409
## beta1   0.603635    0.098756  6.11237 0.000000
## skew    0.871019    0.044760 19.45962 0.000000
## shape   8.688614    2.988168  2.90767 0.003641
##
## LogLikelihood : 968.0014
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9875
## Bayes        -3.9094
## Shibata      -3.9882
## Hannan-Quinn -3.9568
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2488  0.6179
## Lag[2*(p+q)+(p+q)-1][5]    1.1734  0.9999
## Lag[4*(p+q)+(p+q)-1][9]    3.6727  0.7650
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1545  0.6943
## Lag[2*(p+q)+(p+q)-1][8]     2.2433  0.8199
## Lag[4*(p+q)+(p+q)-1][14]    4.7176  0.8036
## d.o.f=3
##
```

```
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]     1.545 0.500 2.000  0.2139
## ARCH Lag[6]     1.701 1.461 1.711  0.5600
## ARCH Lag[8]     1.777 2.368 1.583  0.7861
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.7391
## Individual Statistics:
## ar1    0.16450
## ma1    0.14644
## mxreg1 0.05108
## omega  0.15831
## alpha1 0.27762
## alpha2 0.23423
## beta1  0.22029
## skew   0.45374
## shape  0.29098
##
## 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           0.1161 0.9076
## Negative Sign Bias  0.2991 0.7650
## Positive Sign Bias  0.9886 0.3234
## Joint Effect        2.5312 0.4697
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     15.59       0.6844
## 2    30     30.25       0.4017
## 3    40     47.98       0.1534
## 4    50     47.38       0.5391
##
##
## Elapsed time : 0.315999
```

Second phase of first approach

Next, we extend the GARCH model by including the exogenous variable (lagged S&P500 index) to capture its potential impact on volatility dynamics.

We begin by estimating various ARMA models (without exogenous variables) to explore different lag structures and their impact on the modeling of the log-transformed BlackRock returns.

```r
arma11 = Arima(logbr, order=c(1,0,1), include.mean = F)
summary(arma11)
```

```
## Series: logbr
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##          ar1     ma1
##       0.0185  0.1018
## s.e.  0.3851  0.3826
##
## sigma^2 = 0.001205:  log likelihood = 934.94
## AIC=-1863.88   AICc=-1863.83   BIC=-1851.35
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001619146 0.03464184 0.02572139 126.4757 140.7223 0.7272981
##                      ACF1
## Training set -0.001460161
```

```r
confint(arma11)
```

```
##          2.5 %    97.5 %
## ar1 -0.7363867 0.7733149
## ma1 -0.6479826 0.8516602
```

```r
arma66 = Arima(logbr, order=c(6,0,6),  include.mean = F)
summary(arma66)
```

```
## Series: logbr
## ARIMA(6,0,6) with zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5     ar6      ma1     ma2
##       0.1898  -0.3427  0.1007  -0.2704  0.1164  0.4888  -0.0944  0.3101
## s.e.  0.2413   0.2384  0.2547   0.2328  0.2238  0.2013   0.2236  0.2107
##           ma3     ma4      ma5      ma6
##       -0.0589  0.2598  -0.1305  -0.6347
## s.e.   0.2306  0.2081   0.2105   0.1965
##
## sigma^2 = 0.001172:  log likelihood = 945.7
## AIC=-1865.39   AICc=-1864.61   BIC=-1811.11
##
## Training set error measures:
##                       ME       RMSE        MAE     MPE    MAPE      MASE
## Training set 0.001968132 0.03380102 0.02508844 147.554 206.554 0.7094008
##                    ACF1
## Training set 0.01563397
```

```r
confint(arma66)
```

```
##           2.5 %     97.5 %
## ar1 -0.28303234  0.6627110
## ar2 -0.80997614  0.1246744
## ar3 -0.39849485  0.5999083
## ar4 -0.72675514  0.1859826
## ar5 -0.32228962  0.5550377
## ar6  0.09423391  0.8834042
## ma1 -0.53262955  0.3438621
## ma2 -0.10298368  0.7231270
## ma3 -0.51087245  0.3929916
## ma4 -0.14809923  0.6676742
## ma5 -0.54315246  0.2820824
## ma6 -1.01982896 -0.2496371
```

```
arma16 = Arima(logbr, order=c(1,0,6),  include.mean = F)
summary(arma16)
```

```
## Series: logbr
## ARIMA(1,0,6) with zero mean
##
## Coefficients:
##           ar1     ma1     ma2      ma3      ma4      ma5      ma6
##       -0.1626  0.2793  0.0230  -0.0123  -0.0341  -0.0706  -0.1521
## s.e.   0.2717  0.2684  0.0569   0.0468   0.0475   0.0465   0.0462
##
## sigma^2 = 0.001189:  log likelihood = 940.63
## AIC=-1865.26   AICc=-1864.96   BIC=-1831.86
##
## Training set error measures:
##                       ME      RMSE        MAE      MPE     MAPE      MASE
## Training set 0.002013696 0.0342291 0.02585577 150.6884 190.4817 0.7310978
##                      ACF1
## Training set -0.003678112
```

```
confint(arma16)
```

```
##           2.5 %      97.5 %
## ar1 -0.69523419  0.36996962
## ma1 -0.24662883  0.80528836
## ma2 -0.08848004  0.13455818
## ma3 -0.10402328  0.07935601
## ma4 -0.12715984  0.05904786
## ma5 -0.16167078  0.02042436
## ma6 -0.24267736 -0.06145404
```

```
arma61 = Arima(logbr, order=c(6,0,1),  include.mean = F)
summary(arma61)
```

```
## Series: logbr
## ARIMA(6,0,1) with zero mean
##
## Coefficients:
```

```
##           ar1     ar2      ar3      ar4      ar5      ar6     ma1
##       -0.3051  0.0353  -0.0056  -0.0296  -0.0776  -0.1413  0.4200
## s.e.   0.2665  0.0565   0.0471   0.0470   0.0479   0.0455  0.2676
##
## sigma^2 = 0.001191:  log likelihood = 940.25
## AIC=-1864.51   AICc=-1864.2   BIC=-1831.1
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001928313 0.03425667 0.02583055 153.0041 187.4023 0.7303847
##                      ACF1
## Training set -0.001247587
```

```r
confint(arma61)
```

```
##           2.5 %      97.5 %
## ar1 -0.82748147  0.21729052
## ar2 -0.07553241  0.14613870
## ar3 -0.09793419  0.08670892
## ar4 -0.12168278  0.06238662
## ar5 -0.17143106  0.01632422
## ar6 -0.23047177 -0.05206327
## ma1 -0.10447381  0.94456400
```

```r
ar1 = Arima(logbr, order=c(1,0,0), include.mean = F)
summary(ar1)
```

```
## Series: logbr
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.1195
## s.e.  0.0452
##
## sigma^2 = 0.001203:  log likelihood = 934.91
## AIC=-1865.83   AICc=-1865.8   BIC=-1857.48
##
## Training set error measures:
##                       ME       RMSE        MAE     MPE     MAPE      MASE
## Training set 0.001600652 0.03464359 0.02573681 124.522 138.9352 0.7277341
##                        ACF1
## Training set -0.0006293122
```

```r
confint(ar1)
```

```
##          2.5 %    97.5 %
## ar1 0.03087994 0.2081991
```

```r
predictions11 <- forecast(arma11, h=53)
predictions66 <- forecast(arma66, h=53)
predictions16 <- forecast(arma16,  h=53)
```

```r
predictions61 <- forecast(arma61,  h=53)
predictions1 <- forecast(ar1,  h=53)

rmse11 <- sqrt(mean((predictions11$mean - log_diff_test_br)^2))
rmse66 <- sqrt(mean((predictions66$mean - log_diff_test_br)^2))
rmse16 <- sqrt(mean((predictions16$mean - log_diff_test_br)^2))
rmse61 <- sqrt(mean((predictions61$mean - log_diff_test_br)^2))
rmse1 <- sqrt(mean((predictions1$mean - log_diff_test_br)^2))

print(paste("RMSE for ARIMA(1,0,1):", rmse11))
```

```
## [1] "RMSE for ARIMA(1,0,1): 0.0329268780764259"
```

```r
print(paste("RMSE for ARIMA(6,0,6):", rmse66))
```

```
## [1] "RMSE for ARIMA(6,0,6): 0.0332672739283378"
```

```r
print(paste("RMSE for ARIMA(1,0,6):", rmse16))
```

```
## [1] "RMSE for ARIMA(1,0,6): 0.0329210812226163"
```

```r
print(paste("RMSE for ARIMA(6,0,1):", rmse61))
```

```
## [1] "RMSE for ARIMA(6,0,1): 0.0329021774900373"
```

```r
print(paste("RMSE for ARIMA(1,0,0):", rmse1))
```

```
## [1] "RMSE for ARIMA(1,0,0): 0.0329292276983588"
```

```r
cbind( AIC(ar1), AIC(arma61), AIC(arma16), AIC(arma66), AIC(arma11))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1865.829 -1864.507 -1865.265 -1865.392 -1863.877
```

```r
cbind( BIC(ar1), BIC(arma61), BIC(arma16), BIC(arma66), BIC(arma11))
```

```
##           [,1]    [,2]      [,3]      [,4]     [,5]
## [1,] -1857.477 -1831.1 -1831.858 -1811.106 -1851.35
```

Although the AIC values for all ARMA models are quite similar, the BIC values suggest that the AR(1) model is preferred due to its lower value. However, considering that our primary objective is forecasting, the model with the lowest RMSE, which is ARMA(6,1), is chosen as the final model for its better predictive performance.

We test the adequacy of the ARMA(6,1) model by examining its residuals.

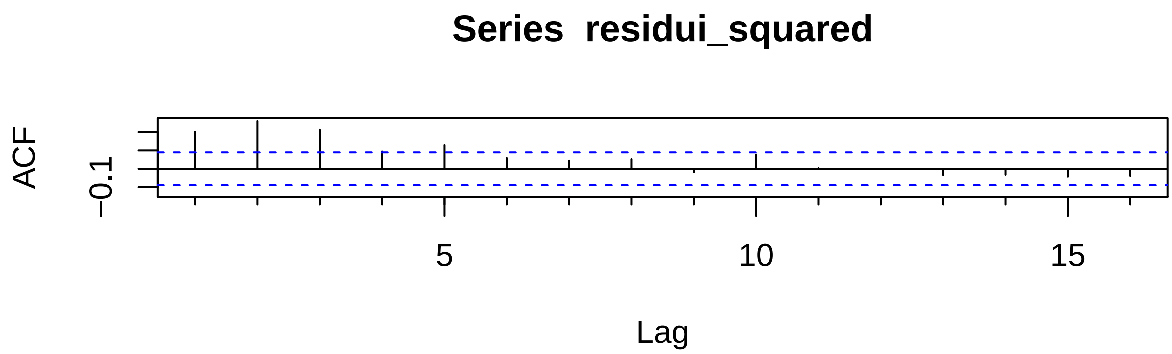
```r
residuals_arima <- arma61$residuals
Box.test(residuals_arima, lag = 12, type = "Ljung")
```

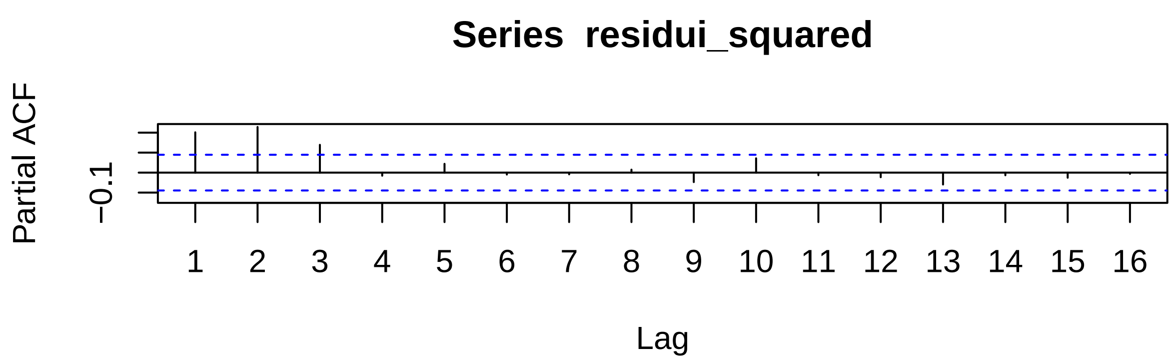
```
##
##  Box-Ljung test
##
## data:  residuals_arima
## X-squared = 4.1626, df = 12, p-value = 0.9803
```

```r
residui_squared <- residuals_arima^2
Box.test(residui_squared, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residui_squared
## X-squared = 94.077, df = 12, p-value = 7.994e-15
```

```r
par(mfrow=c(2,1))
Acf(residui_squared, lag.max = 16)
Pacf(residui_squared, lag.max = 16)
```





```r
par(mfrow=c(1,1))
```

The Ljung-Box test on the residuals of the ARMA(6,0,1) model fails to reject the null hypothesis, indicating no significant autocorrelation, which is a positive result. However, the Ljung-Box test on the squared residuals reveals significant autocorrelation, suggesting the presence of ARCH effects. The ACF and PACF of the squared residuals support the possibility of an ARCH(3,0) or GARCH(1,1) model to capture the volatility dynamics.

```r
g6111x <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1,1),
                        external.regressors = as.matrix(sp_lag)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE),
  distribution.model = "norm" )
garch6111x <- ugarchfit(spec = g6111x, data = logbr)

g6130x <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0),
                        external.regressors = as.matrix(sp_lag)),
  mean.model = list(armaOrder = c(6,1), include.mean = FALSE),
  distribution.model = "norm" )
garch6130x <- ugarchfit(spec = g6130x, data = logbr)

predictions_g6111x <- ugarchforecast(garch6111x, n.ahead = 53,
                             external.forecasts =list(yregfor=test$SP_LAG))
predictions_g6130x <- ugarchforecast(garch6130x, n.ahead = 53,
                            external.forecasts = list(yregfor =test$SP_LAG))

rmse_g6111x <- sqrt(mean((predictions_g6111x@forecast$seriesFor
                          - log_diff_test_br)^2))
rmse_g6130x <- sqrt(mean((predictions_g6130x@forecast$seriesFor
                          - log_diff_test_br)^2))

print(paste("RMSE for ARMA(6,1)-GARCHX(1,1):", rmse_g6111x))
```

```
## [1] "RMSE for ARMA(6,1)-GARCHX(1,1): 0.0329026015222042"
```

```r
print(paste("RMSE for ARMA(6,1)-ARCHX(3,0):", rmse_g6130x))
```

```
## [1] "RMSE for ARMA(6,1)-ARCHX(3,0): 0.0329233715640138"
```

```r
infocriteria(garch6111x)*length(train$SP_LAG)
```

```
##
## Akaike       -1897.327
## Bayes        -1851.297
## Shibata      -1897.817
## Hannan-Quinn -1879.236
```

```r
infocriteria(garch6130x)*length(train$SP_LAG)
```

```
##
## Akaike       -1906.818
## Bayes        -1856.603
## Shibata      -1907.399
## Hannan-Quinn -1887.081
```

The RMSE values for both the ARMA(6,1)-GARCHX(1,1) and ARMA(6,1)-ARCHX(3,0) models are very close, with the GARCH model showing a slightly better RMSE. In terms of model selection criteria, the ARMA(6,1)-ARCHX(3,0) model outperforms the GARCH model based on lower AIC, BIC, and other criteria, making it the preferable choice for capturing volatility dynamics.

```
garch6111x
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.292456    0.288856 -1.01246 0.311317
## ar2     0.027506    0.055974  0.49140 0.623144
## ar3    -0.017293    0.048789 -0.35444 0.723010
## ar4    -0.028299    0.048682 -0.58130 0.561036
## ar5    -0.052791    0.047544 -1.11034 0.266851
## ar6    -0.103955    0.046043 -2.25779 0.023959
## ma1     0.392107    0.285619  1.37284 0.169804
## omega   0.000004    0.000030  0.11990 0.904559
## alpha1  0.129845    0.039680  3.27231 0.001067
## beta1   0.774835    0.021007 36.88388 0.000000
## vxreg1  0.000000    0.000000  0.51069 0.609566
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.292456    0.191438 -1.52768 0.126592
## ar2     0.027506    0.042509  0.64706 0.517595
## ar3    -0.017293    0.045342 -0.38138 0.702921
## ar4    -0.028299    0.041739 -0.67799 0.497779
## ar5    -0.052791    0.043485 -1.21400 0.224748
## ar6    -0.103955    0.045369 -2.29133 0.021944
## ma1     0.392107    0.188356  2.08174 0.037366
## omega   0.000004    0.000024  0.14818 0.882201
## alpha1  0.129845    0.056254  2.30819 0.020989
## beta1   0.774835    0.062229 12.45134 0.000000
## vxreg1  0.000000    0.000006  0.02709 0.978388
##
```

```
## LogLikelihood : 957.6955
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9364
## Bayes        -3.8409
## Shibata      -3.9374
## Hannan-Quinn -3.8988
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1076  0.7429
## Lag[2*(p+q)+(p+q)-1][20]    1.9251  1.0000
## Lag[4*(p+q)+(p+q)-1][34]    8.5317  0.9998
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.330 0.24877
## Lag[2*(p+q)+(p+q)-1][5]     6.384 0.07295
## Lag[4*(p+q)+(p+q)-1][9]     7.648 0.15073
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     1.091 0.500 2.000  0.2962
## ARCH Lag[5]     2.001 1.440 1.667  0.4711
## ARCH Lag[7]     2.103 2.315 1.543  0.6955
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  130.5208
## Individual Statistics:
## ar1      0.16678
## ar2      0.06194
## ar3      0.08486
## ar4      0.11274
## ar5      0.04523
## ar6      0.09877
## ma1      0.15526
## omega    0.14198
## alpha1   0.03762
## beta1    0.06095
## vxreg1 128.74178
##
## 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            1.111 0.26704
## Negative Sign Bias   1.025 0.30586
## Positive Sign Bias   1.308 0.19159
## Joint Effect         6.503 0.08954   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     21.49       0.3101
## 2    30     34.49       0.2218
## 3    40     33.68       0.7109
## 4    50     51.12       0.3904
##
##
## Elapsed time : 0.1590831
```

```
garch6130x
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(3,0)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.399617    0.329660 -1.21221 0.225432
## ar2     0.059926    0.062737  0.95520 0.339474
## ar3    -0.010172    0.052522 -0.19367 0.846438
## ar4    -0.014030    0.047098 -0.29789 0.765787
## ar5    -0.045297    0.044602 -1.01557 0.309832
## ar6    -0.098384    0.044114 -2.23022 0.025733
## ma1     0.511541    0.327363  1.56261 0.118145
## omega   0.000525    0.000080  6.54141 0.000000
## alpha1  0.052910    0.048750  1.08533 0.277777
## alpha2  0.165453    0.051506  3.21227 0.001317
## alpha3  0.170869    0.060567  2.82116 0.004785
## vxreg1  0.000000    0.000000  0.49812 0.618401
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1    -0.399617    0.315037 -1.268475 0.204628
## ar2     0.059926    0.055755  1.074814 0.282458
## ar3    -0.010172    0.048414 -0.210095 0.833593
```

```
## ar4    -0.014030    0.039979 -0.350938 0.725635
## ar5    -0.045297    0.044377 -1.020727 0.307384
## ar6    -0.098384    0.044089 -2.231516 0.025647
## ma1     0.511541    0.308642  1.657393 0.097440
## omega   0.000525    0.000089  5.890777 0.000000
## alpha1  0.052910    0.052993  0.998423 0.318074
## alpha2  0.165453    0.062863  2.631973 0.008489
## alpha3  0.170869    0.065799  2.596822 0.009409
## vxreg1  0.000000    0.000003  0.085475 0.931884
##
## LogLikelihood : 963.431
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9561
## Bayes        -3.8519
## Shibata      -3.9573
## Hannan-Quinn -3.9151
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                    0.003568  0.9524
## Lag[2*(p+q)+(p+q)-1][20]  1.948931  1.0000
## Lag[4*(p+q)+(p+q)-1][34]  9.729070  0.9984
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.02178  0.8827
## Lag[2*(p+q)+(p+q)-1][8]    1.32937  0.9459
## Lag[4*(p+q)+(p+q)-1][14]   4.14750  0.8659
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]    0.1951 0.500 2.000  0.6587
## ARCH Lag[6]    1.0704 1.461 1.711  0.7276
## ARCH Lag[8]    1.2911 2.368 1.583  0.8774
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  65.5765
## Individual Statistics:
## ar1     0.10805
## ar2     0.08869
## ar3     0.09677
## ar4     0.10679
## ar5     0.04472
## ar6     0.14322
```

```
## ma1     0.11514
## omega   0.10909
## alpha1  0.13115
## alpha2  0.23595
## alpha3  0.09471
## vxreg1 57.14925
##
## 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           0.7673 0.4433
## Negative Sign Bias  0.1324 0.8947
## Positive Sign Bias  0.4468 0.6552
## Joint Effect        2.8730 0.4116
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     18.25       0.5057
## 2    30     36.23       0.1668
## 3    40     34.51       0.6748
## 4    50     45.72       0.6071
##
##
## Elapsed time : 0.396879
```

The ARMA(6,1)-GARCHX(1,1) model shows a slight issue with the standardized squared residuals, indicating a potential inadequacy in modeling volatility. Given the minimal difference in RMSE, I prefer the ARMA(6,1)-ARCHX(3,0) model, as it does not exhibit issues with the standardized squared residuals and has lower AIC and BIC values, indicating a better overall fit.

We will try to simplify the model by removing some of the less significant parameters.

```r
g1130x <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0),
                        external.regressors = as.matrix(sp_lag)),
  mean.model = list(armaOrder = c(1,1), include.mean = FALSE),
  distribution.model = "norm"
)

garch1130x <- ugarchfit(spec = g1130x, data = logbr)

predictions_g1130x <- ugarchforecast(garch1130x, n.ahead = 53,
                          external.forecasts = list(yregfor =test$SP_LAG))

rmse_g1130x <- sqrt(mean((predictions_g1130x@forecast$seriesFor
                          - log_diff_test_br)^2))

print(paste("RMSE for ARMA(6,1)-GARCHX(3,0):", rmse_g6130x))
```

```
## [1] "RMSE for ARMA(6,1)-GARCHX(3,0): 0.0329233715640138"
```

```r
print(paste("RMSE for ARMA(1,1)-GARCHX(3,0):", rmse_g1130x))
```

```
## [1] "RMSE for ARMA(1,1)-GARCHX(3,0): 0.0329264827723276"
```

```r
infocriteria(garch6130x )*length(train$SP_LAG)
```

```
##
## Akaike       -1906.818
## Bayes        -1856.603
## Shibata      -1907.399
## Hannan-Quinn -1887.081
```

```r
infocriteria(garch1130x)*length(train$SP_LAG)
```

```
##
## Akaike       -1912.044
## Bayes        -1882.752
## Shibata      -1912.244
## Hannan-Quinn -1900.531
```

The ARMA(1,1)-GARCHX(3,0) has both lower AIC and BIC values, while being more parsimonious in terms of the number of parameters. Therefore, this model is preferred for its better performance and simplicity.

The performance of the model is evaluated using different distributional assumptions for the error terms.

```r
g1130xstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0),
                        external.regressors = as.matrix(sp_lag)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE),
  distribution.model = "std"
)

garch1130xstd <- ugarchfit(spec = g1130xstd, data = logbr)

g1130xsstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0),
                        external.regressors = as.matrix(sp_lag)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE),
  distribution.model = "sstd"
)

garch1130xsstd <- ugarchfit(spec = g1130xsstd, data = logbr)
```

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

```r
#fail to converge

predictions_g1130xstd <- ugarchforecast(garch1130xstd, n.ahead = 53,
                          external.forecasts = list(yregfor =test$SP_LAG))

rmse_g1130xstd <- sqrt(mean((predictions_g1130xstd@forecast$seriesFor
                             - log_diff_test_br)^2))

print(paste("RMSE for ARMA(1,1)-GARCHX(3,0):", rmse_g1130x))
```

```
## [1] "RMSE for ARMA(1,1)-GARCHX(3,0): 0.0329264827723276"
```

```r
print(paste("RMSE for ARMA(1,1)-GARCHX(3,0)_STD:", rmse_g1130xstd))
```

```
## [1] "RMSE for ARMA(1,1)-GARCHX(3,0)_STD: 0.0329278013580711"
```

```r
infocriteria(garch1130x)*length(train$SP_LAG)
```

```
##
## Akaike       -1912.044
## Bayes        -1882.752
## Shibata      -1912.244
## Hannan-Quinn -1900.531
```

```r
infocriteria(garch1130xstd)*length(train$SP_LAG)
```

```
##
## Akaike       -1916.684
## Bayes        -1883.208
## Shibata      -1916.945
## Hannan-Quinn -1903.527
```

The ARMA(1,1)-GARCHX(3,0)-STD model has a slightly lower BIC and AIC compared to the model with normal distribution. The chosen model is ARMA(1,1)-GARCHX(3,0)-STD.

Conclusion of the first approach

In this step, we compare two models: ARMAX(1,1)-GARCH(2,1)-SSTD and ARMA(1,1)-GARCHX(3,0)-STD, both including the lagged S&P500 as an exogenous variable.

```r
p1 <- ugarchforecast(garch11x21sstd, n.ahead = 53,
                     external.forecasts = list(xregfor =test$SP_LAG))
p2 <- ugarchforecast(garch1130xstd, n.ahead = 53,
                     external.forecasts = list(yregfor =test$SP_LAG))

rmse_p1 <- sqrt(mean((p1@forecast$seriesFor - log_diff_test_br)^2))
rmse_p2 <- sqrt(mean((p2@forecast$seriesFor - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD with lagged sp:", rmse_p1))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD with lagged sp: 0.0328996460161713"
```

```r
print(paste("RMSE for ARMA(1,1)-GARCHX(3,0)-STD with lagged sp:", rmse_p2))
```

```
## [1] "RMSE for ARMA(1,1)-GARCHX(3,0)-STD with lagged sp: 0.0329278013580711"
```

```r
infocriteria(garch11x21sstd)*length(train$SP_LAG)
```

```
##
## Akaike       -1921.990
## Bayes        -1884.329
## Shibata      -1922.320
## Hannan-Quinn -1907.188
```

```r
infocriteria(garch1130xstd)*length(train$SP_LAG)
```

```
##
## Akaike       -1916.684
## Bayes        -1883.208
## Shibata      -1916.945
## Hannan-Quinn -1903.527
```

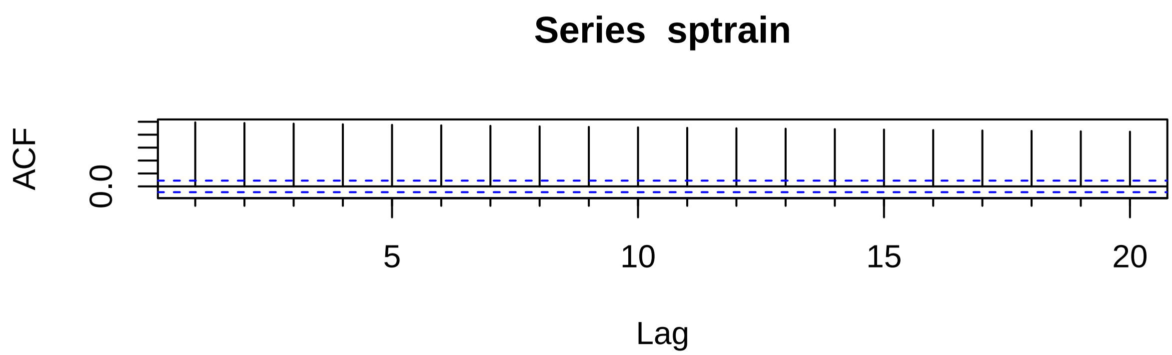
The ARMAX(1,1)-GARCH(2,1)-SSTD model performs slightly better, with lower RMSE, AIC and BIC values, making it the preferred choice for the first approach. Therefore, we select the ARMAX(1,1)-GARCH(2,1)-SSTD model with the lagged S&P500 variable as the chosen model for forecasting.

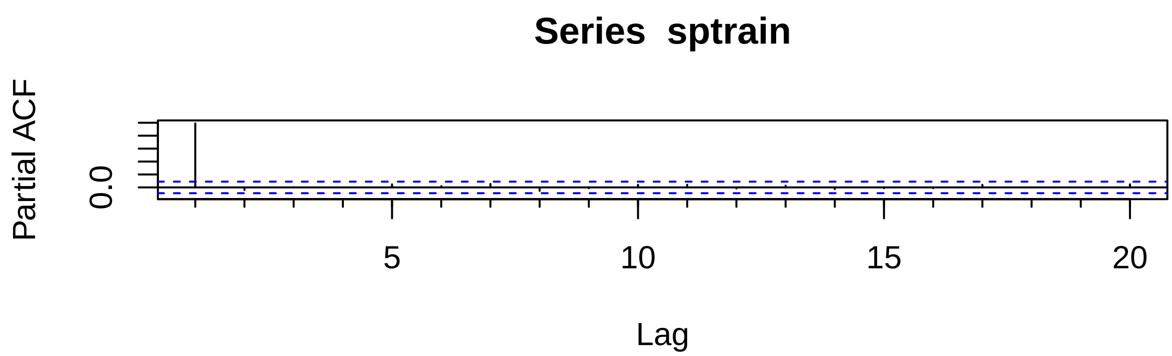
Second approach

Model for the exogenous variable

In the initial phase of this second approach, the analysis is directed towards selecting the most appropriate model for accurately predicting the exogenous variable.

```r
sptrain = log(train$`S&P 500 FIN SVS - PRICE INDEX`)
sptest = log(test$`S&P 500 FIN SVS - PRICE INDEX`)

par(mfrow=c(2,1))
Acf(sptrain, lag.max = 20)
Pacf(sptrain, lag.max=20)
```





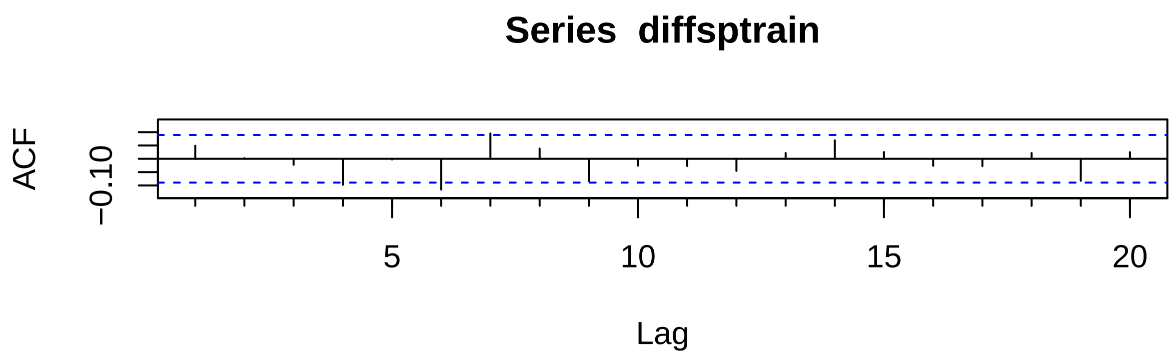

```r
par(mfrow=c(1,1))
```

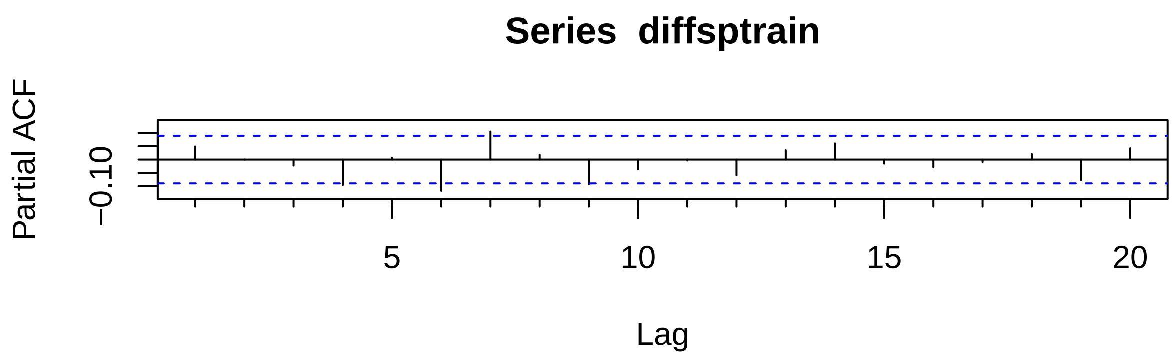
From the ACF and PACF plots of the exogenous variable, we observe a slow decay in the autocorrelations, suggesting non-stationarity.

```r
diffsptrain = diff(sptrain)
t.test(diffsptrain)
```

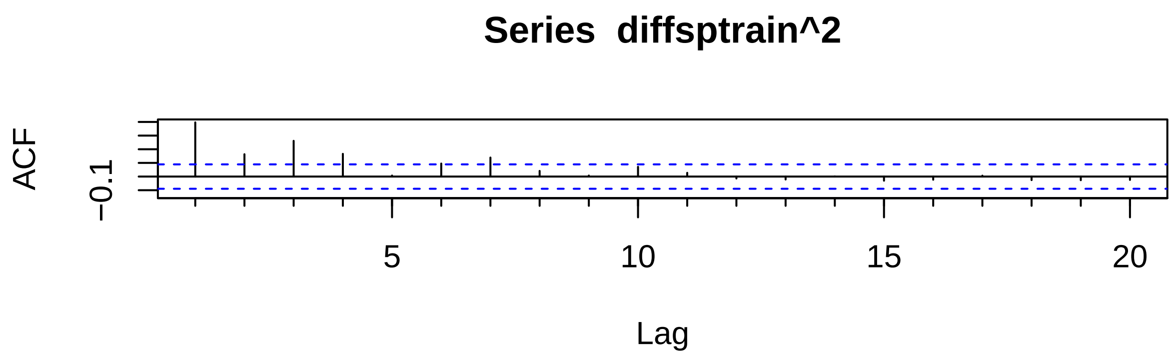
```
##
##  One Sample t-test
##
## data:  diffsptrain
## t = 1.4347, df = 480, p-value = 0.152
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0006612115  0.0042399264
## sample estimates:
##   mean of x
## 0.001789357
```

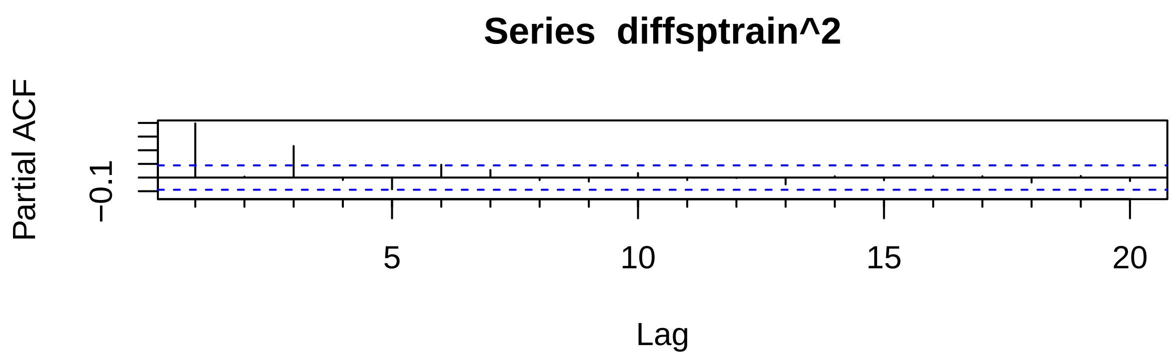
```r
par(mfrow=c(2,1))
Acf(diffsptrain, lag.max = 20)
Pacf(diffsptrain, lag.max=20)
```






```
par(mfrow=c(1,1))

par(mfrow=c(2,1))
Acf(diffsptrain^2, lag.max = 20)
Pacf(diffsptrain^2, lag.max=20)
```






```r
par(mfrow=c(1,1))
```

After differencing, the series appears stationary, as indicated by the ACF and PACF plots. The t-test confirms the absence of a significant mean.

```r
ar(diffsptrain)$order
```

```
## [1] 9
```

```r
ar9 =Arima(diffsptrain, order=c(9,0,0), include.mean = F)
summary(ar9)
```

```
## Series: diffsptrain
## ARIMA(9,0,0) with zero mean
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5      ar6     ar7     ar8
##       0.0658  0.0055  -0.0159  -0.0860  0.0105  -0.1182  0.1082  0.0286
## s.e.  0.0454  0.0455   0.0452   0.0449  0.0451   0.0449  0.0451  0.0454
##          ar9
##       -0.0872
## s.e.   0.0453
##
## sigma^2 = 0.0007302:  log likelihood = 1058.83
```

```
## AIC=-2097.65   AICc=-2097.18   BIC=-2055.89
##
## Training set error measures:
##                       ME       RMSE       MAE      MPE     MAPE      MASE
## Training set 0.001933272 0.02676869 0.0190809 86.15047 170.7567 0.6969691
##                     ACF1
## Training set -0.00806011
```

```r
confint(ar9)
```

```
##            2.5 %       97.5 %
## ar1 -0.02317618  0.154738835
## ar2 -0.08361224  0.094577398
## ar3 -0.10455552  0.072697975
## ar4 -0.17390479  0.001905915
## ar5 -0.07779360  0.098821361
## ar6 -0.20617917 -0.030312428
## ar7  0.01971201  0.196647309
## ar8 -0.06042348  0.117584630
## ar9 -0.17600473  0.001599412
```

```r
ar7 =Arima(diffsptrain, order=c(7,0,0), include.mean = F)
summary(ar7)
```

```
## Series: diffsptrain
## ARIMA(7,0,0) with zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ar4     ar5      ar6     ar7
##       0.0664  -0.0068  -0.0050  -0.0895  0.0178  -0.1179  0.1101
## s.e.  0.0453   0.0451   0.0451   0.0449  0.0451   0.0451  0.0452
##
## sigma^2 = 0.0007332:  log likelihood = 1056.85
## AIC=-2097.71   AICc=-2097.4   BIC=-2064.3
##
## Training set error measures:
##                       ME       RMSE       MAE      MPE     MAPE     MASE
## Training set 0.001829382 0.02688076 0.0191318 62.63406 161.8384 0.698828
##                      ACF1
## Training set -0.007407599
```

```r
confint(ar7)
```

```
##            2.5 %       97.5 %
## ar1 -0.02237895  0.155147552
## ar2 -0.09506771  0.081532500
## ar3 -0.09330629  0.083364445
## ar4 -0.17748278 -0.001582265
## ar5 -0.07060703  0.106129852
## ar6 -0.20622329 -0.029613776
## ar7  0.02143842  0.198722366
```

```r
arma77 = Arima(diffsptrain, order=c(7,0,7), include.mean = F)
summary(arma77)
```

```
## Series: diffsptrain
## ARIMA(7,0,7) with zero mean
##
## Coefficients:
##           ar1      ar2      ar3      ar4     ar5     ar6     ar7     ma1
##       -0.0612  -0.4137  -0.0952  -0.0692  0.0130  0.3131  0.5105  0.0941
## s.e.   0.3115   0.2312   0.3431   0.2958  0.2858  0.2323  0.1799  0.3180
##          ma2     ma3     ma4     ma5      ma6      ma7
##       0.4176  0.1267  0.0348  0.0221  -0.4733  -0.4600
## s.e.  0.2231  0.3348  0.2924  0.2722   0.2361   0.1751
##
## sigma^2 = 0.0007089:  log likelihood = 1065.91
## AIC=-2101.83   AICc=-2100.8   BIC=-2039.19
##
## Training set error measures:
##                        ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001862805 0.02623532 0.01863061 80.27601 220.7311 0.6805213
##                    ACF1
## Training set 0.01397109
```

```r
confint(arma77)
```

```
##           2.5 %      97.5 %
## ar1 -0.67176482  0.54931552
## ar2 -0.86678484  0.03938799
## ar3 -0.76758763  0.57726140
## ar4 -0.64889557  0.51046785
## ar5 -0.54717367  0.57324287
## ar6 -0.14215939  0.76827885
## ar7  0.15794013  0.86298182
## ma1 -0.52917718  0.71743369
## ma2 -0.01963509  0.85475084
## ma3 -0.52945687  0.78292406
## ma4 -0.53815607  0.60785144
## ma5 -0.51140861  0.55566528
## ma6 -0.93592760 -0.01057283
## ma7 -0.80324764 -0.11680204
```

```r
arma62 = Arima(diffsptrain, order=c(6,0,2), include.mean = F)
summary(arma62)
```

```
## Series: diffsptrain
## ARIMA(6,0,2) with zero mean
##
## Coefficients:
##           ar1      ar2     ar3      ar4      ar5      ar6     ma1     ma2
##       -0.4903  -0.8160  0.0314  -0.0926  -0.0789  -0.1762  0.5498  0.8730
## s.e.   0.0685   0.0766  0.0623   0.0622   0.0515   0.0469  0.0555  0.0625
##
```

```
## sigma^2 = 0.0007246:  log likelihood = 1060.06
## AIC=-2102.12   AICc=-2101.73   BIC=-2064.53
##
## Training set error measures:
##                       ME       RMSE        MAE      MPE     MAPE      MASE
## Training set 0.001925327 0.02669421 0.01904891 78.54034 179.9092 0.6958004
##                     ACF1
## Training set 0.002013455
```

```
confint(arma62)
```

```
##            2.5 %      97.5 %
## ar1 -0.62447433 -0.35607845
## ar2 -0.96614842 -0.66588292
## ar3 -0.09072579  0.15355354
## ar4 -0.21445679  0.02927403
## ar5 -0.17986424  0.02210005
## ar6 -0.26813208 -0.08432520
## ma1  0.44099689  0.65861117
## ma2  0.75050122  0.99556027
```

```
cbind(AIC(ar7), AIC(ar9), AIC(arma77), AIC(arma62))
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -2097.706 -2097.652 -2101.828 -2102.115
```

```
cbind(BIC(ar7), BIC(ar9), BIC(arma77), BIC(arma62))
```

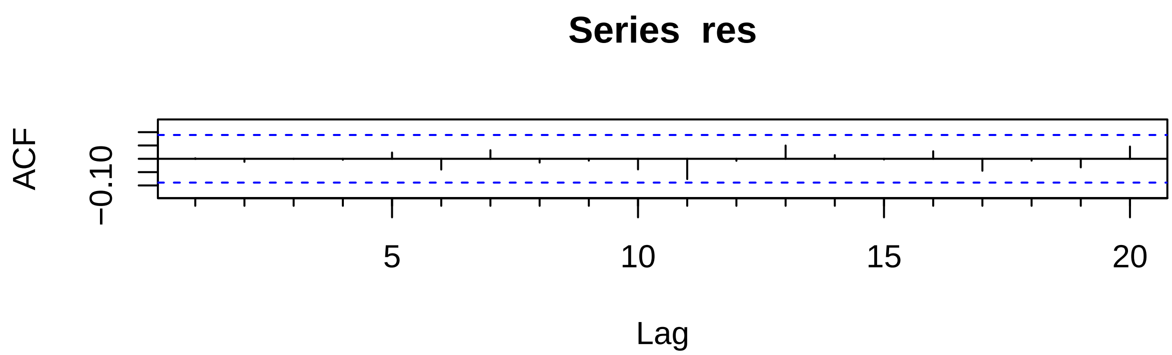
```
##           [,1]      [,2]     [,3]      [,4]
## [1,] -2064.299 -2055.893 -2039.19 -2064.532
```

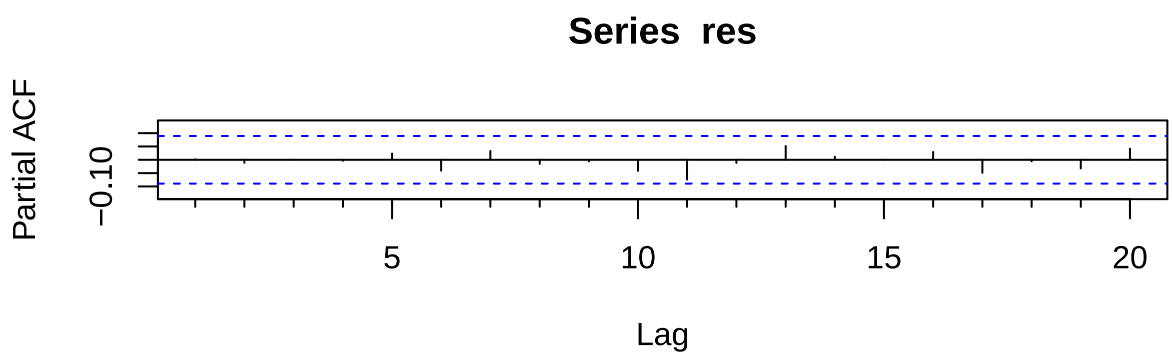
The ARMA(6,2) model has the lowest AIC and BIC, making it the preferred choice for modeling the differenced series.

```
res= arma62$residuals

Box.test(res, lag = 16, fitdf=8)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 7.0069, df = 8, p-value = 0.5359
```

```
par(mfrow=c(2,1))
Acf(res, lag.max = 20)
Pacf(res, lag.max=20)
```

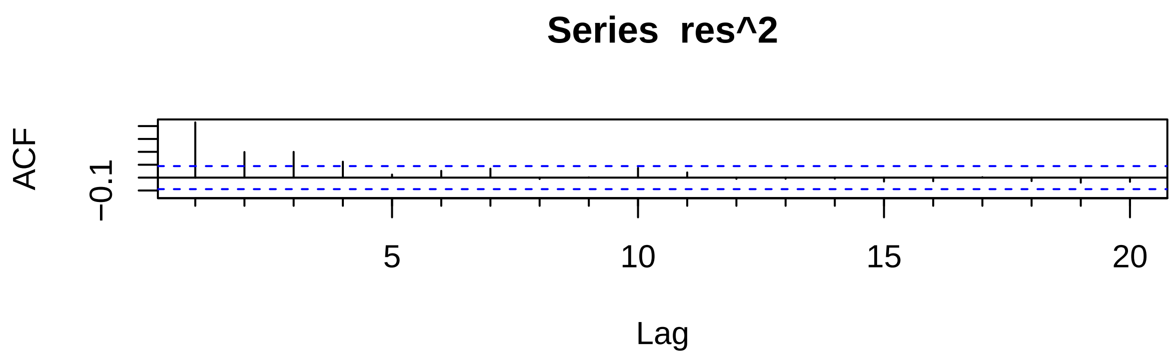


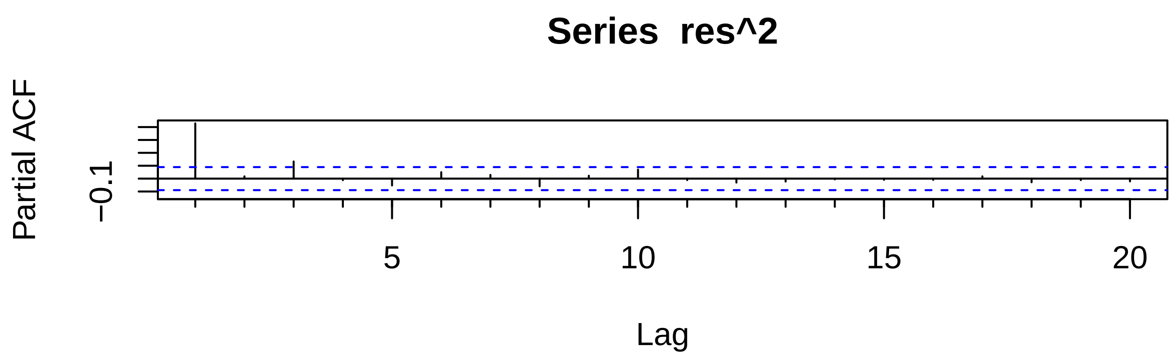



```r
par(mfrow=c(1,1))

Box.test(res^2, lag = 16, fitdf=8)
```

```
##
##  Box-Pierce test
##
## data:  res^2
## X-squared = 142.5, df = 8, p-value < 2.2e-16
```

```r
par(mfrow=c(2,1))
Acf(res^2, lag.max = 20)
Pacf(res^2, lag.max=20)
```






```r
par(mfrow=c(1,1))
```

The residuals of the ARMA(6,2) model show no significant autocorrelation, indicating a good fit in the mean. However, the squared residuals display strong autocorrelation, suggesting the presence of ARCH effects. It seems appropriate to consider either an ARCH(3) or a GARCH(1,1) model for modeling the conditional variance.

```r
mod1 <- garchFit(diffsptrain ~ arma(6,2) + garch(3, 0),
                 data = diffsptrain,  trace = F, include.mean = F)
summary(mod1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(3, 0), data = diffsptrain,
##     include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(3, 0)
## <environment: 0x000001db51fbde38>
##  [data = diffsptrain]
##
## Conditional Distribution:
```

```
##  norm
##
## Coefficient(s):
##        ar1         ar2         ar3         ar4         ar5         ar6
## -0.4773585  -0.4717762   0.0375180   0.0039970  -0.0487647  -0.1505707
##        ma1         ma2       omega      alpha1      alpha2      alpha3
##  0.5229693   0.5263995   0.0003938   0.0742349   0.1671086   0.1725541
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -4.774e-01   1.523e-01   -3.135 0.001720 **
## ar2    -4.718e-01   1.685e-01   -2.799 0.005120 **
## ar3     3.752e-02   6.288e-02    0.597 0.550735
## ar4     3.997e-03   5.672e-02    0.070 0.943824
## ar5    -4.876e-02   4.955e-02   -0.984 0.325020
## ar6    -1.506e-01   4.037e-02   -3.730 0.000192 ***
## ma1     5.230e-01   1.519e-01    3.444 0.000574 ***
## ma2     5.264e-01   1.656e-01    3.179 0.001478 **
## omega   3.938e-04   4.329e-05    9.098  < 2e-16 ***
## alpha1  7.423e-02   4.093e-02    1.814 0.069697 .
## alpha2  1.671e-01   4.816e-02    3.470 0.000521 ***
## alpha3  1.726e-01   7.572e-02    2.279 0.022672 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1097.036    normalized:  2.280741
##
## Description:
##  Tue May 20 20:32:20 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  91.3533236 0.000000e+00
##  Shapiro-Wilk Test  R    W       0.9668571 5.897154e-09
##  Ljung-Box Test     R    Q(10)   1.5452565 9.987874e-01
##  Ljung-Box Test     R    Q(15)   3.5392487 9.988987e-01
##  Ljung-Box Test     R    Q(20)   6.4135391 9.982112e-01
##  Ljung-Box Test     R^2  Q(10)   8.6587901 5.647625e-01
##  Ljung-Box Test     R^2  Q(15)   9.1111069 8.716367e-01
##  Ljung-Box Test     R^2  Q(20)  12.5631636 8.953308e-01
##  LM Arch Test       R    TR^2    9.4716735 6.622022e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.511586 -4.407406 -4.512791 -4.470638
```

```r
mod1t <- garchFit(diffsptrain ~ arma(6,2) + garch(3, 0),
                  data = diffsptrain,  trace = F, include.mean = F,
                  cond.dist = "std")
```

```r
summary(mod1t)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(3, 0), data = diffsptrain,
##     cond.dist = "std", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(3, 0)
## <environment: 0x000001db4f5b97a8>
##  [data = diffsptrain]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ar5          ar6
## -0.42796754  -0.47208125   0.01634934  -0.01221148  -0.04557628  -0.11940920
##         ma1          ma2        omega       alpha1       alpha2       alpha3
##  0.45986581   0.49215384   0.00041803   0.08191643   0.13177820   0.20623695
##       shape
##  4.83724606
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -4.280e-01   1.799e-01   -2.379  0.01737 *
## ar2    -4.721e-01   2.259e-01   -2.090  0.03660 *
## ar3     1.635e-02   5.735e-02    0.285  0.77559
## ar4    -1.221e-02   5.215e-02   -0.234  0.81485
## ar5    -4.558e-02   4.569e-02   -0.998  0.31847
## ar6    -1.194e-01   4.369e-02   -2.733  0.00628 **
## ma1     4.599e-01   1.826e-01    2.519  0.01178 *
## ma2     4.922e-01   2.242e-01    2.195  0.02815 *
## omega   4.180e-04   7.061e-05    5.921 3.21e-09 ***
## alpha1  8.192e-02   6.256e-02    1.309  0.19040
## alpha2  1.318e-01   6.995e-02    1.884  0.05958 .
## alpha3  2.062e-01   1.040e-01    1.984  0.04730 *
## shape   4.837e+00   1.176e+00    4.113 3.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1110.838    normalized:  2.309434
##
## Description:
##  Tue May 20 20:32:22 2025 by user: User
##
```

```
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  127.7893345 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9616929 7.235340e-10
##  Ljung-Box Test     R    Q(10)    2.6960320 9.877009e-01
##  Ljung-Box Test     R    Q(15)    4.5606648 9.952379e-01
##  Ljung-Box Test     R    Q(20)    7.9550724 9.921608e-01
##  Ljung-Box Test     R^2  Q(10)   11.5466235 3.165440e-01
##  Ljung-Box Test     R^2  Q(15)   12.2044911 6.634893e-01
##  Ljung-Box Test     R^2  Q(20)   15.1836943 7.658018e-01
##  LM Arch Test       R    TR^2    12.4997430 4.064238e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.564814 -4.451953 -4.566225 -4.520455
```

```
mod1st <- garchFit(diffsptrain ~ arma(6,2) + garch(3, 0),
                   data = diffsptrain,  trace = F, include.mean = F,
                   cond.dist = "sstd")
summary(mod1st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(3, 0), data = diffsptrain,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(3, 0)
## <environment: 0x000001db5b5362a8>
##  [data = diffsptrain]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ar5          ar6
## -0.35024191  -0.46201018  -0.02746592  -0.06430650  -0.08172696  -0.11889656
##         ma1          ma2        omega       alpha1       alpha2       alpha3
##  0.35549270   0.41536683   0.00048165   0.09847834   0.12995666   0.19824594
##        skew        shape
##  0.76371846   4.43672778
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -3.502e-01   2.446e-01   -1.432   0.1522
## ar2    -4.620e-01   2.056e-01   -2.247   0.0247 *
## ar3    -2.747e-02   5.574e-02   -0.493   0.6222
```

```
## ar4    -6.431e-02   5.053e-02   -1.273   0.2032
## ar5    -8.173e-02   4.385e-02   -1.864   0.0624 .
## ar6    -1.189e-01   4.650e-02   -2.557   0.0106 *
## ma1     3.555e-01   2.485e-01    1.430   0.1526
## ma2     4.154e-01   2.052e-01    2.024   0.0429 *
## omega   4.817e-04   8.722e-05    5.522 3.35e-08 ***
## alpha1  9.848e-02   6.660e-02    1.479   0.1393
## alpha2  1.300e-01   7.176e-02    1.811   0.0701 .
## alpha3  1.982e-01   9.745e-02    2.034   0.0419 *
## skew    7.637e-01   4.797e-02   15.921  < 2e-16 ***
## shape   4.437e+00   1.105e+00    4.015 5.95e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1120.996    normalized:  2.330553
##
## Description:
##  Tue May 20 20:32:23 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  174.0987410 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9554353 7.122972e-11
##  Ljung-Box Test     R    Q(10)    5.6981755 8.399518e-01
##  Ljung-Box Test     R    Q(15)    7.4571850 9.437006e-01
##  Ljung-Box Test     R    Q(20)   11.0857232 9.439691e-01
##  Ljung-Box Test     R^2  Q(10)   12.0017629 2.849385e-01
##  Ljung-Box Test     R^2  Q(15)   12.5002220 6.408394e-01
##  Ljung-Box Test     R^2  Q(20)   15.6270937 7.394726e-01
##  LM Arch Test       R    TR^2    12.6033666 3.985161e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.602893 -4.481351 -4.604525 -4.555122
```

```r
mod1@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC
## -2170.073 -2119.962 -2170.652 -2150.377
```

```r
mod1t@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC
## -2195.676 -2141.389 -2196.354 -2174.339
```

```r
mod1st@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC
## -2213.992 -2155.530 -2214.776 -2191.014
```

Given that the residuals for all three models (mod1, mod1t, mod1st) show similar issues, particularly with non-normality (which is common in financial time series), the key distinguishing factor for model selection remains the comparison of AIC and BIC. Since mod1st has the lowest AIC and BIC, it is chosen as the most parsimonious model.

```r
mod2 <- garchFit(diffsptrain ~ arma(6,2) + garch(1,1),
                 data = diffsptrain,  trace = F, include.mean = F,
                 cond.dist = "norm")
summary(mod2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(1, 1), data = diffsptrain,
##     cond.dist = "norm", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(1, 1)
## <environment: 0x000001db543bd2d8>
##  [data = diffsptrain]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ar5          ar6
## -0.46637053  -0.43261340   0.03919063  -0.02513860  -0.05516519  -0.15203934
##         ma1          ma2        omega       alpha1        beta1
##  0.53727459   0.49451629   0.00010433   0.14235983   0.70259010
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -4.664e-01   1.719e-01   -2.713 0.006677 **
## ar2    -4.326e-01   1.869e-01   -2.315 0.020638 *
## ar3     3.919e-02   6.041e-02    0.649 0.516507
## ar4    -2.514e-02   5.741e-02   -0.438 0.661483
## ar5    -5.517e-02   5.392e-02   -1.023 0.306278
## ar6    -1.520e-01   4.516e-02   -3.366 0.000761 ***
## ma1     5.373e-01   1.736e-01    3.095 0.001971 **
## ma2     4.945e-01   1.870e-01    2.644 0.008194 **
## omega   1.043e-04   3.557e-05    2.933 0.003360 **
## alpha1  1.424e-01   3.783e-02    3.763 0.000168 ***
## beta1   7.026e-01   7.277e-02    9.655  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1095.78    normalized:  2.278129
##
```

```
## Description:
##  Tue May 20 20:32:23 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  161.2398805 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9604558 4.493784e-10
##  Ljung-Box Test     R    Q(10)    1.6976607 9.981762e-01
##  Ljung-Box Test     R    Q(15)    3.4344425 9.990804e-01
##  Ljung-Box Test     R    Q(20)    6.5694823 9.978765e-01
##  Ljung-Box Test     R^2  Q(10)   14.0797739 1.693839e-01
##  Ljung-Box Test     R^2  Q(15)   15.3284712 4.280258e-01
##  Ljung-Box Test     R^2  Q(20)   18.2885156 5.684087e-01
##  LM Arch Test       R    TR^2    15.9343760 1.942603e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.510520 -4.415022 -4.511535 -4.472985
```

```r
mod2t =  garchFit(diffsptrain ~ arma(6,2) + garch(1,1),
                  data = diffsptrain,  trace = F, include.mean = F,
                  cond.dist = "std")
summary(mod2t)  # problemi arch effect
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(1, 1), data = diffsptrain,
##     cond.dist = "std", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(1, 1)
## <environment: 0x000001db514a3348>
##  [data = diffsptrain]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ar5          ar6
## -4.0640e-01  -3.1736e-01   8.3754e-03  -2.1600e-02  -3.6020e-02  -1.1673e-01
##         ma1          ma2        omega       alpha1        beta1        shape
##  4.4456e-01   3.3185e-01   4.3638e-05   9.8671e-02   8.4811e-01   4.3668e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -4.064e-01   2.218e-01   -1.832   0.0669 .
## ar2    -3.174e-01   2.536e-01   -1.251   0.2108
```

```
## ar3     8.375e-03   4.998e-02    0.168   0.8669
## ar4    -2.160e-02   5.012e-02   -0.431   0.6665
## ar5    -3.602e-02   4.720e-02   -0.763   0.4453
## ar6    -1.167e-01   4.854e-02   -2.405   0.0162 *
## ma1     4.446e-01   2.267e-01    1.961   0.0498 *
## ma2     3.319e-01   2.545e-01    1.304   0.1923
## omega   4.364e-05   2.991e-05    1.459   0.1445
## alpha1  9.867e-02   4.480e-02    2.202   0.0276 *
## beta1   8.481e-01   7.157e-02   11.850  < 2e-16 ***
## shape   4.367e+00   1.018e+00    4.290 1.79e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1114.017    normalized:  2.316044
##
## Description:
##  Tue May 20 20:32:24 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  279.7814463 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9502516 1.217050e-11
##  Ljung-Box Test     R    Q(10)    3.3538918 9.718281e-01
##  Ljung-Box Test     R    Q(15)    4.7116740 9.942989e-01
##  Ljung-Box Test     R    Q(20)    8.6711788 9.864123e-01
##  Ljung-Box Test     R^2  Q(10)   20.3334514 2.625196e-02
##  Ljung-Box Test     R^2  Q(15)   23.3695722 7.660217e-02
##  Ljung-Box Test     R^2  Q(20)   25.7755416 1.733537e-01
##  LM Arch Test       R    TR^2    21.5457187 4.293805e-02
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.582192 -4.478013 -4.583397 -4.541245
```

```r
mod2st =  garchFit(diffsptrain ~ arma(6,2) + garch(1,1),
                  data = diffsptrain,  trace = F, include.mean = F,
                  cond.dist = "sstd")
summary(mod2st)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = diffsptrain ~ arma(6, 2) + garch(1, 1), data = diffsptrain,
##     cond.dist = "sstd", include.mean = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(6, 2) + garch(1, 1)
## <environment: 0x000001db5234ba50>
##  [data = diffsptrain]
##
```

```
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ar5          ar6
## -2.6708e-01  -3.3466e-01  -3.3259e-02  -8.1752e-02  -8.9590e-02  -1.1930e-01
##         ma1          ma2        omega       alpha1        beta1         skew
##  2.6726e-01   2.7052e-01   4.8354e-05   1.0752e-01   8.5603e-01   7.3216e-01
##       shape
##  3.9345e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -2.671e-01   2.830e-01   -0.944   0.3452
## ar2    -3.347e-01   2.304e-01   -1.453   0.1463
## ar3    -3.326e-02   4.976e-02   -0.668   0.5039
## ar4    -8.175e-02   5.037e-02   -1.623   0.1046
## ar5    -8.959e-02   4.520e-02   -1.982   0.0475 *
## ar6    -1.193e-01   5.162e-02   -2.311   0.0208 *
## ma1     2.673e-01   2.876e-01    0.929   0.3527
## ma2     2.705e-01   2.299e-01    1.177   0.2394
## omega   4.835e-05   2.878e-05    1.680   0.0929 .
## alpha1  1.075e-01   4.701e-02    2.287   0.0222 *
## beta1   8.560e-01   6.091e-02   14.055  < 2e-16 ***
## skew    7.322e-01   5.015e-02   14.600  < 2e-16 ***
## shape   3.935e+00   9.422e-01    4.176 2.97e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1126.292    normalized:  2.341564
##
## Description:
##  Tue May 20 20:32:25 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                   Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  355.0861762 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9423242 1.017277e-12
##  Ljung-Box Test     R    Q(10)    8.1455732 6.146200e-01
##  Ljung-Box Test     R    Q(15)    9.1978749 8.669459e-01
##  Ljung-Box Test     R    Q(20)   12.9635049 8.789444e-01
##  Ljung-Box Test     R^2  Q(10)   19.3789115 3.570565e-02
##  Ljung-Box Test     R^2  Q(15)   22.0211935 1.072494e-01
##  Ljung-Box Test     R^2  Q(20)   24.1896090 2.342078e-01
##  LM Arch Test       R    TR^2    19.5865672 7.532214e-02
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.629075 -4.516214 -4.630485 -4.584715
```

```r
mod2@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC 
## -2169.560 -2123.626 -2170.048 -2151.506
```

```r
mod2t@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC 
## -2204.034 -2153.924 -2204.614 -2184.339
```

```r
mod2st@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC 
## -2226.585 -2172.299 -2227.263 -2205.248
```

Mod2st has the lowest AIC and BIC among the models, suggesting it is the most parsimonious choice. However, its summary reveals issues with the presence of ARCH effects in the residuals, which indicates that the model may not fully capture volatility clustering. Similar residual problems are also present in mod2t.

```r
mod1st@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC 
## -2213.992 -2155.530 -2214.776 -2191.014
```

```r
mod2@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC 
## -2169.560 -2123.626 -2170.048 -2151.506
```

The final choice between mod1st and mod2 is primarily based on the AIC and BIC criteria. Both models do not show significant issues with the residuals, but mod1st has lower AIC and BIC values compared to mod2, indicating a better fit to the time series. Therefore, mod1st (ARMA(6,2)-GARCH(3,0)) is selected for its superior efficiency in terms of parsimony and performance.

We now attempt to simplify the model by reducing the number of parameters and evaluating if a more parsimonious model can provide similar or better performance.

```r
mod5st =  garchFit(diffsptrain ~ arma(4,2) + garch(3,0),
                   data = diffsptrain,  trace = F, include.mean = F,
                   cond.dist = "sstd")
summary(mod5st)
```

```
## 
## Title:
##  GARCH Modelling 
## 
## Call:
##  garchFit(formula = diffsptrain ~ arma(4, 2) + garch(3, 0), data = diffsptrain, 
##     cond.dist = "sstd", include.mean = F, trace = F)
```

```
##
## Mean and Variance Equation:
##  data ~ arma(4, 2) + garch(3, 0)
## <environment: 0x000001db52d55388>
##  [data = diffsptrain]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##         ar1          ar2          ar3          ar4          ma1          ma2
##  0.60411715  -0.62213208   0.06201115  -0.11341305  -0.60996421   0.58662022
##       omega       alpha1       alpha2       alpha3         skew        shape
##  0.00048478   0.09797359   0.10577888   0.23443250   0.76809999   4.25568779
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1     6.041e-01   1.928e-01    3.133  0.00173 **
## ar2    -6.221e-01   1.216e-01   -5.118 3.09e-07 ***
## ar3     6.201e-02   5.511e-02    1.125  0.26048
## ar4    -1.134e-01   4.299e-02   -2.638  0.00834 **
## ma1    -6.100e-01   1.933e-01   -3.155  0.00161 **
## ma2     5.866e-01   1.173e-01    4.999 5.75e-07 ***
## omega   4.848e-04   9.236e-05    5.249 1.53e-07 ***
## alpha1  9.797e-02   6.709e-02    1.460  0.14420
## alpha2  1.058e-01   6.759e-02    1.565  0.11756
## alpha3  2.344e-01   1.080e-01    2.171  0.02991 *
## skew    7.681e-01   4.296e-02   17.880  < 2e-16 ***
## shape   4.256e+00   1.044e+00    4.077 4.57e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1119.991    normalized:  2.328463
##
## Description:
##  Tue May 20 20:32:26 2025 by user: User
##
##
## Standardised Residuals Tests:
##                                  Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  212.9211155 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9500505 1.139236e-11
##  Ljung-Box Test     R    Q(10)   11.9538017 2.881603e-01
##  Ljung-Box Test     R    Q(15)   13.1112593 5.937046e-01
##  Ljung-Box Test     R    Q(20)   17.7158900 6.061177e-01
##  Ljung-Box Test     R^2  Q(10)   13.2757112 2.086624e-01
##  Ljung-Box Test     R^2  Q(15)   14.0122941 5.245969e-01
##  Ljung-Box Test     R^2  Q(20)   16.1810133 7.053329e-01
##  LM Arch Test       R    TR^2    13.9662820 3.028666e-01
##
```

```
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.607030 -4.502851 -4.608235 -4.566083
```

```r
mod1st@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC
## -2213.992 -2155.530 -2214.776 -2191.014
```

```r
mod5st@fit$ics * length(diffsptrain)
```

```
##       AIC       BIC       SIC      HQIC
## -2215.982 -2165.871 -2216.561 -2196.286
```

The simplified model, ARMA(4,2)-GARCH(3,0)-SSTD, shows lower AIC and BIC values compared to the initial model, making it a more parsimonious choice. This suggests that the simplified model retains predictive performance while improving efficiency. Therefore, we select ARMA(4,2)-ARCH(3,0)-SSTD model for predicting the S&P 500 index.

Model for BlackRosck series

We now proceed to model the series logbr, this time using the aligned (non-lagged) exogenous variable, similarly to the first approach. Based on the previously analyzed ACF and PACF plots of logbr, we begin by identifying an appropriate ARIMAX structure for the mean equation, and will subsequently assess the need for a GARCH component to model the conditional variance, if necessary.

```r
train <- data[1:train_size, ]
test <- data[(train_size+1):nrow(data), ]
sp= train$`S&P 500 FIN SVS - PRICE INDEX` [-1]
logbr = diff(log(train$BLACKROCK))
length(sp)==length(logbr)
```

```
## [1] TRUE
```

```r
armax11 = Arima(logbr, order=c(1,0,1), xreg=sp, include.mean = F)
confint(armax11)
```

```
##              2.5 %       97.5 %
## ar1  -7.546874e-01 7.965869e-01
## ma1  -6.694640e-01 8.657790e-01
## xreg -9.169993e-05 9.678851e-05
```

```r
armax66 = Arima(logbr, order=c(6,0,6), xreg=sp, include.mean = F)
confint(armax66)
```

```
## Warning in sqrt(diag(vcov(object))): Si è prodotto un NaN
```

```
##           2.5 %       97.5 %
## ar1  -0.1664572  0.566081561
## ar2  -0.6733618  0.015063192
## ar3  -0.3297380  0.553479159
## ar4  -0.6046379  0.100579753
## ar5  -0.2452854  0.484992950
## ar6   0.1980050  0.813799942
## ma1  -0.3242290  0.108064346
## ma2         NaN          NaN
## ma3  -0.3508431  0.205379150
## ma4         NaN          NaN
## ma5  -0.2799344  0.001793565
## ma6  -0.7472989 -0.564244022
## xreg        NaN          NaN
```

```r
armax16 = Arima(logbr, order=c(1,0,6), xreg=sp, include.mean = F)
confint(armax16)
```

```
##              2.5 %       97.5 %
## ar1  -0.7033367230  0.404030586
## ma1  -0.2908130371  0.819183701
## ma2  -0.1060215233  0.140995325
## ma3  -0.1158520093  0.084679713
## ma4  -0.1384425526  0.062436554
## ma5  -0.1712861047  0.020658026
## ma6  -0.2518367785 -0.058379566
## xreg -0.0001179362  0.000122522
```

```r
armax61 = Arima(logbr, order=c(6,0,1), xreg=sp, include.mean = F)
confint(armax61)
```

```
##              2.5 %        97.5 %
## ar1  -8.293620e-01  0.2250902752
## ar2  -8.292806e-02  0.1472627965
## ar3  -1.018920e-01  0.0868496823
## ar4  -1.264728e-01  0.0626294486
## ar5  -1.767257e-01  0.0143102886
## ar6  -2.351575e-01 -0.0525259043
## ma1  -1.168634e-01  0.9477990173
## xreg -9.806402e-05  0.0001027276
```

```r
ar1 = Arima(logbr, order=c(1,0,0), xreg=sp, include.mean = F)
confint(ar1)
```

```
##              2.5 %       97.5 %
## ar1   2.657096e-02 0.2104176159
## xreg -9.019147e-05 0.0000952772
```

```r
cbind(AIC(ar1), AIC(armax61), AIC(armax16), AIC(armax66), AIC(armax11))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1869.509 -1868.763 -1869.619 -1869.602 -1867.579
```

```r
cbind(BIC(ar1), BIC(armax61), BIC(armax16), BIC(armax66), BIC(armax11))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1856.975 -1831.161 -1832.017 -1811.111 -1850.867
```

Although AIC values are quite similar across models, the ARMAX(1,0) model has the lowest BIC, indicating it as the preferred choice due to its greater parsimony.

```r
sp_test = test$`S&P 500 FIN SVS - PRICE INDEX`[-1]

predictions11 <- forecast(armax11, xreg=sp_test, h=53)
predictions66 <- forecast(armax66, xreg=sp_test, h=53)
predictions16 <- forecast(armax16, xreg=sp_test, h=53)
predictions61 <- forecast(armax61, xreg=sp_test, h=53)
predictions1 <- forecast(ar1, xreg=sp_test, h=53)

rmse11 <- sqrt(mean((predictions11$mean - log_diff_test_br)^2))
rmse66 <- sqrt(mean((predictions66$mean - log_diff_test_br)^2))
rmse16 <- sqrt(mean((predictions16$mean - log_diff_test_br)^2))
rmse61 <- sqrt(mean((predictions61$mean - log_diff_test_br)^2))
rmse1 <- sqrt(mean((predictions1$mean - log_diff_test_br)^2))

print(paste("RMSE for ARIMAX(1,0,1):", rmse11))
```

```
## [1] "RMSE for ARIMAX(1,0,1): 0.0329070581349692"
```

```r
print(paste("RMSE for ARIMAX(6,0,6):", rmse66))
```

```
## [1] "RMSE for ARIMAX(6,0,6): 0.0332195965078546"
```

```r
print(paste("RMSE for ARIMAX(1,0,6):", rmse16))
```

```
## [1] "RMSE for ARIMAX(1,0,6): 0.0328776796660606"
```

```r
print(paste("RMSE for ARIMAX(6,0,1):", rmse61))
```

```
## [1] "RMSE for ARIMAX(6,0,1): 0.0328645073642815"
```

```r
print(paste("RMSE for ARIMAX(1,0,0):", rmse1))
```

```
## [1] "RMSE for ARIMAX(1,0,0): 0.0329079468067389"
```

Although ARMAX(1,0) shows good performance with an RMSE of 0.03291, the ARMAX(6,1) model achieves a slightly lower RMSE of 0.03286. Therefore, despite being more complex, we choose ARMAX(6,1) for its better forecasting accuracy.

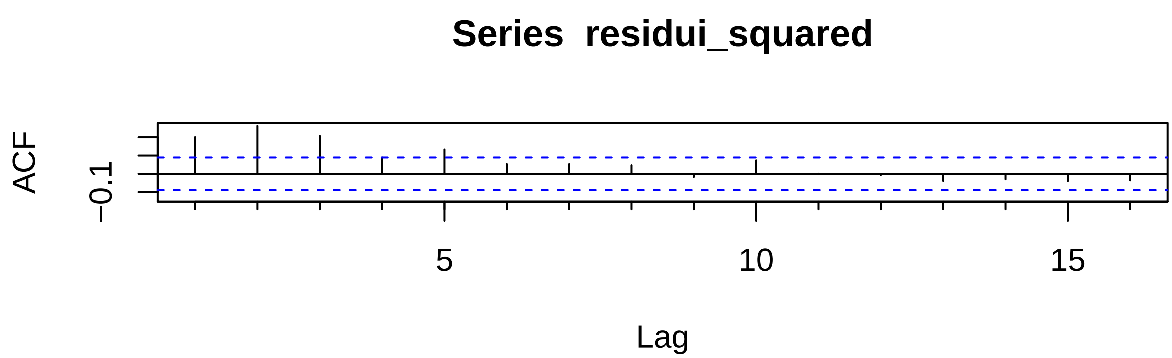
```r
residuals_arimax <- armax61$residuals
Box.test(residuals_arimax, lag = 12, type = "Ljung")
```

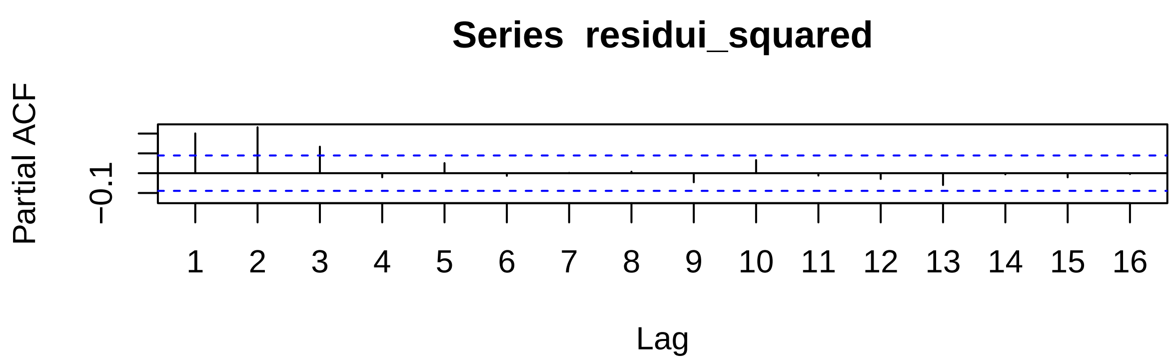
```
##
##  Box-Ljung test
##
## data:  residuals_arimax
## X-squared = 3.9805, df = 12, p-value = 0.9838
```

```r
residui_squared <- residuals_arimax^2
Box.test(residui_squared, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residui_squared
## X-squared = 93.552, df = 12, p-value = 1.01e-14
```

```r
par(mfrow=c(2,1))
Acf(residui_squared, lag.max = 16)
Pacf(residui_squared, lag.max = 16)
```






```r
par(mfrow=c(1,1))
```

The residuals from the ARIMAX(6,0,1) model do not show significant autocorrelation, indicating a good fit in the mean equation. However, the squared residuals exhibit strong autocorrelation, suggesting the presence of conditional heteroskedasticity, thus motivating the use of a GARCH-type model. The ACF and PACF suggest that GARCH(1,1) or ARCH(3,0) models may be appropriate.

```r
gg61x11 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "norm" )
garchg61x11 <- ugarchfit(spec = gg61x11, data = logbr)

gg61x30 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(3,0)),
  mean.model = list(armaOrder = c(6,1), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "norm" )
garchg61x30 <- ugarchfit(spec = gg61x30, data = logbr)

predictions_gg61x11 <- ugarchforecast(garchg61x11, n.ahead = 53,
                          external.forecasts = list(mregfor =sp_test))
predictions_gg61x30 <- ugarchforecast(garchg61x30, n.ahead = 53,
                          external.forecasts = list(mregfor =sp_test))

rmse_gg61x11 <- sqrt(mean((predictions_gg61x11@forecast$seriesFor
                           - log_diff_test_br)^2))
rmse_gg61x30 <- sqrt(mean((predictions_gg61x30@forecast$seriesFor
                           - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(6,1)-GARCH(1,1):", rmse_gg61x11))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1): 0.0329154589200727"
```

```r
print(paste("RMSE for ARMAX(6,1)-ARCH(3):", rmse_gg61x30))
```

```
## [1] "RMSE for ARMAX(6,1)-ARCH(3): 0.0330258322891886"
```

```r
infocriteria(garchg61x11)*length(train$BLACKROCK)
```

```
##
## Akaike       -1907.447
## Bayes        -1861.394
## Shibata      -1907.935
## Hannan-Quinn -1889.347
```

```r
infocriteria(garchg61x30)*length(train$BLACKROCK)
```

```
##
## Akaike       -1863.797
## Bayes        -1813.558
## Shibata      -1864.377
## Hannan-Quinn -1844.053
```

The model ARMAX(6,1)-GARCH(1,1) shows a lower RMSE, AIC, and BIC compared to the ARMAX(6,1)-ARCH(3) model. Given these metrics, the ARMAX(6,1)-GARCH(1,1) is selected as the preferred model for further analysis, as it offers better predictive accuracy and model efficiency.

```
garchg61x11
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.229611    0.282507 -0.81276 0.416353
## ar2     0.023010    0.056353  0.40831 0.683046
## ar3    -0.019905    0.048070 -0.41408 0.678814
## ar4    -0.030208    0.047613 -0.63445 0.525790
## ar5    -0.054227    0.046424 -1.16808 0.242774
## ar6    -0.112593    0.045892 -2.45340 0.014151
## ma1     0.329194    0.282223  1.16643 0.243440
## mxreg1  0.000003    0.000002  1.70880 0.087489
## omega   0.000197    0.000072  2.74449 0.006061
## alpha1  0.139777    0.044126  3.16769 0.001537
## beta1   0.688488    0.086778  7.93394 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.229611    0.190690 -1.20411 0.228548
## ar2     0.023010    0.044666  0.51515 0.606448
## ar3    -0.019905    0.045197 -0.44040 0.659645
## ar4    -0.030208    0.040556 -0.74485 0.456363
## ar5    -0.054227    0.042727 -1.26915 0.204389
## ar6    -0.112593    0.044278 -2.54287 0.010995
## ma1     0.329194    0.192800  1.70744 0.087740
## mxreg1  0.000003    0.000002  1.59780 0.110087
## omega   0.000197    0.000066  2.98866 0.002802
## alpha1  0.139777    0.045383  3.07997 0.002070
## beta1   0.688488    0.079303  8.68170 0.000000
##
## LogLikelihood : 962.7487
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9492
## Bayes        -3.8538
## Shibata      -3.9502
## Hannan-Quinn -3.9117
##
## Weighted Ljung-Box Test on Standardized Residuals
```

```
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1966  0.6575
## Lag[2*(p+q)+(p+q)-1][20]    2.3281  1.0000
## Lag[4*(p+q)+(p+q)-1][34]   10.0312  0.9975
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.508 0.21952
## Lag[2*(p+q)+(p+q)-1][5]     7.680 0.03537
## Lag[4*(p+q)+(p+q)-1][9]     9.171 0.07495
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.9267 0.500 2.000  0.3357
## ARCH Lag[5]    1.9709 1.440 1.667  0.4777
## ARCH Lag[7]    2.1761 2.315 1.543  0.6800
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1092
## Individual Statistics:
## ar1    0.12654
## ar2    0.07463
## ar3    0.08935
## ar4    0.11479
## ar5    0.04203
## ar6    0.09242
## ma1    0.11203
## mxreg1 0.08079
## omega  0.17454
## alpha1 0.22075
## beta1  0.18692
##
## 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           1.2373 0.2166
## Negative Sign Bias  1.0377 0.2999
## Positive Sign Bias  0.9514 0.3419
## Joint Effect        5.5232 0.1373
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
```

```
##   group statistic p-value(g-1)
## 1    20     17.75      0.53912
## 2    30     35.84      0.17820
## 3    40     51.44      0.08758
## 4    50     57.83      0.18135
##
##
## Elapsed time : 0.2713649
```

```
garchg61x30
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(3,0)
## Mean Model   : ARFIMA(6,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## ar1     1.092535    0.039006  2.8010e+01 0.000000
## ar2    -0.103652    0.022261 -4.6561e+00 0.000003
## ar3    -0.012493    0.061485 -2.0318e-01 0.838991
## ar4    -0.010047    0.066471 -1.5115e-01 0.879861
## ar5    -0.042929    0.067424 -6.3670e-01 0.524323
## ar6     0.052119    0.040051  1.3013e+00 0.193147
## ma1    -1.000000    0.000000 -4.7952e+06 0.000000
## mxreg1  0.000003    0.000000  8.3109e+00 0.000000
## omega   0.001182    0.000078  1.5197e+01 0.000000
## alpha1  0.000000    0.018834  0.0000e+00 1.000000
## alpha2  0.000000    0.012277  2.0000e-06 0.999999
## alpha3  0.000000    0.017325  1.0000e-06 0.999999
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## ar1     1.092535    0.058632  1.8634e+01  0.00000
## ar2    -0.103652    0.075449 -1.3738e+00  0.16950
## ar3    -0.012493    0.060613 -2.0611e-01  0.83671
## ar4    -0.010047    0.054878 -1.8307e-01  0.85474
## ar5    -0.042929    0.052880 -8.1181e-01  0.41690
## ar6     0.052119    0.038395  1.3575e+00  0.17464
## ma1    -1.000000    0.000000 -2.7038e+06  0.00000
## mxreg1  0.000003    0.000000  6.2936e+00  0.00000
## omega   0.001182    0.000156  7.5543e+00  0.00000
## alpha1  0.000000    0.018001  0.0000e+00  1.00000
## alpha2  0.000000    0.008712  2.0000e-06  1.00000
## alpha3  0.000000    0.015291  1.0000e-06  1.00000
##
## LogLikelihood : 941.9692
```

```
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.8588
## Bayes        -3.7548
## Shibata      -3.8600
## Hannan-Quinn -3.8179
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.03162  0.8589
## Lag[2*(p+q)+(p+q)-1][20]   6.74403  1.0000
## Lag[4*(p+q)+(p+q)-1][34]  16.10044  0.6565
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic   p-value
## Lag[1]                       17.06 3.617e-05
## Lag[2*(p+q)+(p+q)-1][8]      74.97 0.000e+00
## Lag[4*(p+q)+(p+q)-1][14]     85.81 0.000e+00
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##              Statistic Shape Scale   P-Value
## ARCH Lag[4]      5.533 0.500 2.000 0.0186656
## ARCH Lag[6]     16.388 1.461 1.711 0.0002319
## ARCH Lag[8]     19.362 2.368 1.583 0.0001378
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  12.7224
## Individual Statistics:
## ar1    0.11365
## ar2    0.09589
## ar3    0.08525
## ar4    0.08292
## ar5    0.09281
## ar6    0.13278
## ma1    0.10287
## mxreg1 0.20447
## omega  0.72233
## alpha1 0.61775
## alpha2 1.39181
## alpha3 1.19375
##
## 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            0.6772 0.498611
## Negative Sign Bias  2.8050 0.005238 ***
## Positive Sign Bias  0.5585 0.576779
## Joint Effect       15.4342 0.001481 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     47.88    0.0002674
## 2    30     59.37    0.0007395
## 3    40     60.90    0.0139500
## 4    50     77.54    0.0057911
##
##
## Elapsed time : 0.9703109
```

The second model, ARMAX(6,1)-ARCH(3), exhibits significant ARCH effects in its residuals, indicating that it may not adequately capture the volatility dynamics in the data. On the other hand, the first model, ARMAX(6,1)-GARCH(1,1), while not entirely free from issues, shows much less severe problems in the standardized residuals, making it the preferable choice despite its slight imperfections.

```r
gg61x11std <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "std"
)

garchg61x11std <- ugarchfit(spec = gg61x11std, data = logbr)

predictions_gg61x11std <- ugarchforecast(garchg61x11std, n.ahead = 53,
                          external.forecasts = list(mregfor =sp_test))
rmse_gg61x11std <- sqrt(mean((predictions_gg61x11std@forecast$seriesFor
                              - log_diff_test_br)^2))

gg61x11sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
  mean.model = list(armaOrder = c(6, 1), include.mean = FALSE,
                   external.regressors = as.matrix(sp)),
  distribution.model = "sstd"
)

garchg61x11sstd <- ugarchfit(spec = gg61x11sstd, data = logbr)

predictions_gg61x11sstd <- ugarchforecast(garchg61x11sstd, n.ahead = 53,
                              external.forecasts = list(mregfor =sp_test))
rmse_gg61x11sstd <- sqrt(mean((predictions_gg61x11sstd@forecast$seriesFor
                               - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(6,1)-GARCH(1,1):", rmse_gg61x11))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1): 0.0329154589200727"
```

```r
print(paste("RMSE for ARMAX(6,1)-GARCH(1,1)-STD:", rmse_gg61x11std))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1)-STD: 0.0330904210721329"
```

```r
print(paste("RMSE for ARMAX(6,1)-GARCH(1,1)-SSTD:", rmse_gg61x11sstd))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1)-SSTD: 0.032899879778355"
```

```r
infocriteria(garchg61x11)*length(train$BLACKROCK)
```

```
##
## Akaike       -1907.447
## Bayes        -1861.394
## Shibata      -1907.935
## Hannan-Quinn -1889.347
```

```r
infocriteria(garchg61x11std)*length(train$BLACKROCK)
```

```
##
## Akaike       -1917.451
## Bayes        -1867.211
## Shibata      -1918.030
## Hannan-Quinn -1897.706
```

```r
infocriteria(garchg61x11sstd)*length(train$BLACKROCK)
```

```
##
## Akaike       -1922.687
## Bayes        -1868.261
## Shibata      -1923.365
## Hannan-Quinn -1901.297
```

The ARMAX(6,1)-GARCH(1,1)-SSTD model outperforms the others with the lowest RMSE, and the best AIC and BIC values. While the GARCH(1,1) model has a slightly higher RMSE, the GARCH(1,1)-STD model shows improvements in RMSE but lags behind the SSTD version in AIC and BIC. Therefore, the ARMAX(6,1)-GARCH(1,1)-SSTD model is the preferred choice.

```r
garchg61x11sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
```

```
## Mean Model   : ARFIMA(6,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.108630    0.317513 -0.34213 0.732254
## ar2    -0.029449    0.055257 -0.53295 0.594067
## ar3    -0.016471    0.047549 -0.34641 0.729033
## ar4    -0.038534    0.044290 -0.87004 0.384279
## ar5    -0.082644    0.045752 -1.80635 0.070864
## ar6    -0.097014    0.049254 -1.96965 0.048878
## ma1     0.191392    0.318510  0.60090 0.547908
## mxreg1  0.000003    0.000002  1.86537 0.062130
## omega   0.000155    0.000067  2.30582 0.021121
## alpha1  0.136809    0.045674  2.99532 0.002742
## beta1   0.728525    0.085054  8.56546 0.000000
## skew    0.835864    0.056867 14.69859 0.000000
## shape   8.888664    3.820841  2.32636 0.019999
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.108630    0.173453 -0.62628 0.531132
## ar2    -0.029449    0.045798 -0.64302 0.520209
## ar3    -0.016471    0.046678 -0.35288 0.724182
## ar4    -0.038534    0.039511 -0.97527 0.329426
## ar5    -0.082644    0.041830 -1.97572 0.048186
## ar6    -0.097014    0.047842 -2.02782 0.042579
## ma1     0.191392    0.182326  1.04972 0.293847
## mxreg1  0.000003    0.000002  1.60426 0.108656
## omega   0.000155    0.000053  2.95384 0.003138
## alpha1  0.136809    0.039740  3.44259 0.000576
## beta1   0.728525    0.068002 10.71325 0.000000
## skew    0.835864    0.050784 16.45929 0.000000
## shape   8.888664    3.310220  2.68522 0.007248
##
## LogLikelihood : 972.3532
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9807
## Bayes        -3.8680
## Shibata      -3.9821
## Hannan-Quinn -3.9364
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7148  0.3979
## Lag[2*(p+q)+(p+q)-1][20]    4.1795  1.0000
## Lag[4*(p+q)+(p+q)-1][34]   11.5536  0.9831
## d.o.f=7
## H0 : No serial correlation
```

```
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.566 0.21076
## Lag[2*(p+q)+(p+q)-1][5]     7.572 0.03759
## Lag[4*(p+q)+(p+q)-1][9]     8.916 0.08456
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     0.306 0.500 2.000  0.5801
## ARCH Lag[5]     1.271 1.440 1.667  0.6546
## ARCH Lag[7]     1.474 2.315 1.543  0.8265
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.0522
## Individual Statistics:
## ar1    0.29308
## ar2    0.11818
## ar3    0.07053
## ar4    0.07519
## ar5    0.07289
## ar6    0.06281
## ma1    0.24799
## mxreg1 0.08478
## omega  0.13534
## alpha1 0.19991
## beta1  0.15520
## skew   0.57029
## shape  0.25365
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.89 3.15 3.69
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.9277 0.3540
## Negative Sign Bias  0.9937 0.3209
## Positive Sign Bias  1.2290 0.2197
## Joint Effect        5.2532 0.1542
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     15.93       0.6623
## 2    30     27.38       0.5513
## 3    40     41.49       0.3629
## 4    50     44.56       0.6537
```

```
##
##
## Elapsed time : 0.4271128
```

The model selected performs well in terms of RMSE and information criteria, but the standardize squared residuals are not perfect, indicating some room for improvement. To address this, we attempt a more complex GARCH component while simplifying the ARMA part by removing non-significant terms.

```
gg22x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd" )

garchg22x21sstd <- ugarchfit(spec = gg22x21sstd, data = logbr)

predictions_gg22x21sstd <- ugarchforecast(garchg22x21sstd, n.ahead = 53,
                            external.forecasts = list(mregfor =sp_test))
rmse_gg22x21sstd <- sqrt(mean((predictions_gg22x21sstd@forecast$seriesFor
                               - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(6,1)-GARCH(1,1)_SSTD:", rmse_gg61x11sstd))
```

```
## [1] "RMSE for ARMAX(6,1)-GARCH(1,1)_SSTD: 0.032899879778355"
```

```
print(paste("RMSE for ARMAX(2,2)-GARCH(2,1)-SSTD:", rmse_gg22x21sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-GARCH(2,1)-SSTD: 0.0329661385788098"
```

```
infocriteria(garchg61x11sstd)*length(train$BLACKROCK)
```

```
##
## Akaike       -1922.687
## Bayes        -1868.261
## Shibata      -1923.365
## Hannan-Quinn -1901.297
```

```
infocriteria(garchg22x21sstd)*length(train$BLACKROCK)
```

```
##
## Akaike       -1927.452
## Bayes        -1881.399
## Shibata      -1927.940
## Hannan-Quinn -1909.353
```

The first model has a slightly lower RMSE, indicating better predictive performance. However, the second model does not exhibit issues with standardize squared residuals and has lower AIC and BIC values. Since the RMSE difference is minimal, we opt for the second model (ARMAX(2,2)-GARCH(2,1)-SSTD) due to its better overall fit and more stable residuals.

```
garchg22x21sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     1.087028    0.203178  5.35013 0.000000
## ar2    -0.258400    0.202995 -1.27294 0.203040
## ma1    -0.993784    0.208194 -4.77336 0.000002
## ma2     0.129687    0.209223  0.61985 0.535357
## mxreg1  0.000004    0.000002  2.32848 0.019887
## omega   0.000222    0.000097  2.28140 0.022524
## alpha1  0.041718    0.052966  0.78763 0.430911
## alpha2  0.143684    0.069870  2.05645 0.039740
## beta1   0.620487    0.125231  4.95473 0.000001
## skew    0.856137    0.058114 14.73199 0.000000
## shape   9.253357    4.089701  2.26260 0.023660
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     1.087028    0.121666  8.93450 0.000000
## ar2    -0.258400    0.128440 -2.01184 0.044237
## ma1    -0.993784    0.128210 -7.75120 0.000000
## ma2     0.129687    0.129722  0.99973 0.317440
## mxreg1  0.000004    0.000002  1.97613 0.048140
## omega   0.000222    0.000080  2.75962 0.005787
## alpha1  0.041718    0.049257  0.84694 0.397027
## alpha2  0.143684    0.066371  2.16484 0.030400
## beta1   0.620487    0.105502  5.88128 0.000000
## skew    0.856137    0.051288 16.69263 0.000000
## shape   9.253357    3.315057  2.79131 0.005249
##
## LogLikelihood : 972.7306
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9906
## Bayes        -3.8952
## Shibata      -3.9916
## Hannan-Quinn -3.9531
##
## Weighted Ljung-Box Test on Standardized Residuals
```

```
## ------------------------------------
##                           statistic p-value
## Lag[1]                       0.4648  0.4954
## Lag[2*(p+q)+(p+q)-1][11]     3.3232  1.0000
## Lag[4*(p+q)+(p+q)-1][19]     5.1551  0.9922
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                           statistic p-value
## Lag[1]                       0.1782  0.6729
## Lag[2*(p+q)+(p+q)-1][8]      2.1361  0.8374
## Lag[4*(p+q)+(p+q)-1][14]     4.7825  0.7959
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##              Statistic Shape Scale P-Value
## ARCH Lag[4]     1.191 0.500 2.000  0.2752
## ARCH Lag[6]     1.692 1.461 1.711  0.5624
## ARCH Lag[8]     1.872 2.368 1.583  0.7673
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.9186
## Individual Statistics:
## ar1    0.03404
## ar2    0.02617
## ma1    0.04115
## ma2    0.02920
## mxreg1 0.08165
## omega  0.13518
## alpha1 0.20095
## alpha2 0.22094
## beta1  0.19708
## skew   0.57655
## shape  0.27118
##
## 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          1.73971 0.08255   *
## Negative Sign Bias 0.41771 0.67635
## Positive Sign Bias 0.03686 0.97061
## Joint Effect       5.77095 0.12330
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
```

```
##   group statistic p-value(g-1)
## 1    20     13.68       0.8018
## 2    30     21.78       0.8293
## 3    40     33.68       0.7106
## 4    50     50.57       0.4112
##
##
## Elapsed time : 0.451431
```

The chosen model is ARMAX(2,2)-GARCH(2,1)-SSTD. Additionally, we observe that the exogenous variable is significant in the model.

Forecast

In this analysis, we applied two approaches for forecasting. Initially, the data was divided into training and testing sets to select the most appropriate models for each approach. However, now we are using all available data together to make the one-step-ahead forecasts, ensuring that the most up-to-date information is incorporated into the forecast for each approach.

```r
# fist approach
data = read_excel(here('blackrock.xlsx'), sheet = 5, col_names = T)[c(1,2,3,4)]
```

```
## New names:
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...8`
## * `` -> `...9`
```

```r
logbr = diff(log(data$BLACKROCK))
last_logbr <- tail(log(data$BLACKROCK),1)
last_sp_lag <- tail(data$`S&P 500 FIN SVS - PRICE INDEX`, 1)
data <- data %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
sp_lag <- data$SP_LAG[-1]

g11x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd" )
garch11x21sstd <- ugarchfit(spec = g11x21sstd, data =logbr)

forecast <- ugarchforecast(garch11x21sstd, n.ahead = 1,
                  external.forecasts = list(xregfor =last_sp_lag))
predicted_log_returns <- forecast@forecast$seriesFor
predicted_price <- exp(last_logbr + predicted_log_returns)
cat("Prediction of the price for the next week (first approach): ", predicted_price, "\n")
```

```
## Prediction of the price for the next week (first approach):  893.3836
```

```r
# second approach
data = read_excel(here('blackrock.xlsx'), sheet = 5, col_names = T)[c(1,2,3,4)]
```

```
## New names:
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...8`
## * `` -> `...9`
```

```r
sp= data$`S&P 500 FIN SVS - PRICE INDEX` [-1]
logbr = diff(log(data$BLACKROCK))

gg22x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd" )
garchg22x21sstd <- ugarchfit(spec = gg22x21sstd, data = logbr)

last_log_sp <- log(tail(data$`S&P 500 FIN SVS - PRICE INDEX`, 1))
mod3 =  garchFit(diff(log(data$`S&P 500 FIN SVS - PRICE INDEX`)) ~
                 arma(4,2) + garch(3,0), data =
                diff(log(data$`S&P 500 FIN SVS - PRICE INDEX`)) ,
                trace = F, include.mean = F, cond.dist = "sstd")
forsp = predict(mod3, n.ahead = 1)
predicted_log_sp <- last_log_sp + forsp$meanForecast
predicted_sp <- exp(predicted_log_sp)
forecast2 <- ugarchforecast(garchg22x21sstd, n.ahead = 1,
                      external.forecasts = list(xregfor =predicted_sp))
predicted_log_returns2 <- forecast2@forecast$seriesFor
predicted_price2 <- exp(last_logbr + predicted_log_returns2)
cat("Prediction of the price for the next week (second approach):", predicted_price2, "\n")
```

```
## Prediction of the price for the next week (second approach): 895.8519
```

```r
cat("Last observed value of external variable (used in the first
    approach with lag): 1363.83")
```

```
## Last observed value of external variable (used in the first
##     approach with lag): 1363.83
```

```r
cat("Predicted value of external variable (used in the second
    approach with exogenous variable aligned):", predicted_sp, "\n")
```

```
## Predicted value of external variable (used in the second
##     approach with exogenous variable aligned): 1368.169
```

- First Approach (with lagged exogenous variable): The model used is ARMAX(1,1) - GARCH(2,1) - SSTD, where the exogenous variable is lagged. The predicted price is approximately 893.38, using the last observed value of the exogenous variable (S&P 500) as a lagged input.

- Second Approach (with aligned exogenous variable): The model used is ARMAX(2,2) - GARCH(2,1) - SSTD, where the exogenous variable is aligned and estimated through its own ARMA(4,2) - GARCH(3,0) - SSTD model. The predicted price is approximately 895.85, with a slight increase in the predicted value of the exogenous variable (from 1363.83 to 1368.169).

```
predicted_price1 <- 893.3836
predicted_price2 <- 895.8519
real_price <- 888.4399

abs_error1 <- abs(real_price - predicted_price1)
abs_error2 <- abs(real_price - predicted_price2)

perc_error1 <- (abs_error1 / real_price) * 100
perc_error2 <- (abs_error2 / real_price) * 100

cat("Absolute error (first approach):", abs_error1, "\n")
```

```
## Absolute error (first approach): 4.9437
```

```
cat("Percentage error (first approach):", perc_error1, "%\n\n")
```

```
## Percentage error (first approach): 0.5564473 %
```

```
cat("Absolute error (second approach):", abs_error2, "\n")
```

```
## Absolute error (second approach): 7.412
```

```
cat("Percentage error (second approach):", perc_error2, "%\n")
```

```
## Percentage error (second approach): 0.8342714 %
```

The absolute error for the first approach is 4.94, with a percentage error of 0.56%. For the second approach, the absolute error is slightly higher at 7.41, with a percentage error of 0.83%. Although the second approach produces a slightly higher error, both models perform well, with relatively small errors in predicting the price of BlackRock.

WEEK 6

Introduction

We are revisiting the code from last week to re-estimate the two final models chosen on the updated data series in order to assess how to improve the fit of the models.

```
data = read_excel(here('blackrock.xlsx'), sheet = 6, col_names = T)
logbr = diff(log(data$BLACKROCK))
last_logbr <- tail(log(data$BLACKROCK),1)
last_sp_lag <- tail(data$`S&P 500 FIN SVS - PRICE INDEX`, 1)
data <- data %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
```

```r
sp_lag <- data$SP_LAG[-1]

g11x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd"
)
garch11x21sstd <- ugarchfit(spec = g11x21sstd, data =logbr)

# 2
data = read_excel(here('blackrock.xlsx'), sheet = 6, col_names = T)
sp= data$`S&P 500 FIN SVS - PRICE INDEX` [-1]
logbr = diff(log(data$BLACKROCK))
length(sp)==length(logbr)
```

```
## [1] TRUE
```

```r
gg22x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE, external.regressors = as.matrix(sp)),
  distribution.model = "sstd"
)

garchg22x21sstd <- ugarchfit(spec = gg22x21sstd, data = logbr)

garch11x21sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.323590    0.907282 -0.35666 0.721347
## ma1     0.383875    0.888007  0.43229 0.665532
## mxreg1  0.000002    0.000002  1.24469 0.213245
## omega   0.000245    0.000103  2.36991 0.017793
## alpha1  0.054994    0.050107  1.09754 0.272404
## alpha2  0.119544    0.063495  1.88274 0.059735
## beta1   0.615133    0.125547  4.89963 0.000001
## skew    0.862472    0.053505 16.11943 0.000000
## shape  13.179483    7.698567  1.71194 0.086908
##
```

```
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.323590    1.233832 -0.26227 0.793117
## ma1     0.383875    1.210990  0.31699 0.751249
## mxreg1  0.000002    0.000002  1.22398 0.220959
## omega   0.000245    0.000081  3.00613 0.002646
## alpha1  0.054994    0.045174  1.21738 0.223461
## alpha2  0.119544    0.060467  1.97702 0.048040
## beta1   0.615133    0.099054  6.21010 0.000000
## skew    0.862472    0.044556 19.35712 0.000000
## shape  13.179483    6.722690  1.96045 0.049943
##
## LogLikelihood : 1077.695
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9728
## Bayes        -3.9011
## Shibata      -3.9734
## Hannan-Quinn -3.9448
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3039  0.5815
## Lag[2*(p+q)+(p+q)-1][5]    1.1681  0.9999
## Lag[4*(p+q)+(p+q)-1][9]    4.5908  0.5494
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       0.168  0.6819
## Lag[2*(p+q)+(p+q)-1][8]      2.215  0.8245
## Lag[4*(p+q)+(p+q)-1][14]     4.240  0.8564
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]    0.9719 0.500 2.000  0.3242
## ARCH Lag[6]    1.0448 1.461 1.711  0.7349
## ARCH Lag[8]    1.1357 2.368 1.583  0.9034
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.2486
## Individual Statistics:
## ar1    0.20473
## ma1    0.20964
## mxreg1 0.04228
## omega  0.27594
```

```
## alpha1 0.31362
## alpha2 0.21543
## beta1  0.31307
## skew   0.39859
## shape  0.74072
##
## 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           1.6564 0.09823   *
## Negative Sign Bias  0.4963 0.61991
## Positive Sign Bias  0.2322 0.81644
## Joint Effect        5.7197 0.12608
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     15.01       0.7219
## 2    30     28.43       0.4950
## 3    40     35.83       0.6153
## 4    50     46.01       0.5949
##
##
## Elapsed time : 0.3290319
```

```
garchg22x21sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     1.240320    0.098022  12.6535 0.000000
## ar2    -0.371785    0.115008  -3.2327 0.001226
## ma1    -1.185480    0.105486 -11.2383 0.000000
## ma2     0.287323    0.087372   3.2885 0.001007
## mxreg1  0.000003    0.000001   2.0847 0.037098
## omega   0.000221    0.000098   2.2435 0.024865
## alpha1  0.050837    0.050021   1.0163 0.309487
## alpha2  0.109135    0.062741   1.7395 0.081953
```

```
## beta1   0.646206    0.122835   5.2608 0.000000
## skew    0.844313    0.053619  15.7465 0.000000
## shape  13.480114    7.966426   1.6921 0.090624
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     1.240320    0.154130   8.0472 0.000000
## ar2    -0.371785    0.198462  -1.8733 0.061022
## ma1    -1.185480    0.189223  -6.2650 0.000000
## ma2     0.287323    0.144124   1.9936 0.046198
## mxreg1  0.000003    0.000002   1.4097 0.158620
## omega   0.000221    0.000081   2.7136 0.006656
## alpha1  0.050837    0.044817   1.1343 0.256659
## alpha2  0.109135    0.059217   1.8430 0.065332
## beta1   0.646206    0.105081   6.1496 0.000000
## skew    0.844313    0.047150  17.9071 0.000000
## shape  13.480114    7.205922   1.8707 0.061387
##
## LogLikelihood : 1080.101
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9744
## Bayes        -3.8867
## Shibata      -3.9752
## Hannan-Quinn -3.9401
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.5397  0.4625
## Lag[2*(p+q)+(p+q)-1][11]    4.3976  0.9982
## Lag[4*(p+q)+(p+q)-1][19]    6.8490  0.9209
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1562  0.6927
## Lag[2*(p+q)+(p+q)-1][8]     2.0013  0.8587
## Lag[4*(p+q)+(p+q)-1][14]    4.2232  0.8582
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]    0.6294 0.500 2.000  0.4276
## ARCH Lag[6]    0.9461 1.461 1.711  0.7633
## ARCH Lag[8]    1.1694 2.368 1.583  0.8979
##
## Nyblom stability test
## ------------------------------------
```

```
## Joint Statistic:  2.2381
## Individual Statistics:
## ar1    0.02760
## ar2    0.03047
## ma1    0.03758
## ma2    0.03277
## mxreg1 0.06405
## omega  0.24950
## alpha1 0.23687
## alpha2 0.20808
## beta1  0.29060
## skew   0.45320
## shape  0.67383
##
## 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          2.07547 0.03842  **
## Negative Sign Bias 0.52785 0.59782
## Positive Sign Bias 0.04085 0.96743
## Joint Effect       7.89035 0.04833  **
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     13.08       0.8346
## 2    30     24.19       0.7193
## 3    40     36.42       0.5880
## 4    50     47.13       0.5492
##
##
## Elapsed time : 0.490711
```

The Sign Bias Test evaluates whether the volatility dynamics of the GARCH models correctly capture the asymmetric effects of shocks—i.e., whether positive and negative shocks of equal magnitude have different impacts on volatility.

In both models, the overall joint effect from the Sign Bias Test is marginally significant or close to significance, with the first model showing a borderline individual sign bias. The second model displays a clearly significant individual and joint sign bias, suggesting potential misspecification in how asymmetric shocks are captured. This motivates the exploration of alternative specifications, such as TGARCH or EGARCH, to better account for asymmetries in volatility dynamics.

```r
data = read_excel(here('blackrock.xlsx'), sheet = 6, col_names = T)
train_size <- floor(0.9 * nrow(data))
train <- data[1:train_size, ]
test <- data[(train_size+1):nrow(data), ]

br = train$BLACKROCK
logbr = diff(log(br))
```

Specifically, we estimate three alternative models:

- An standard GARCH model, which serves as a baseline symmetric volatility specification.
- An EGARCH model, which captures potential leverage effects and allows for asymmetric responses of volatility to past shocks.
- An TGARCH model, which is another asymmetric volatility model designed to capture threshold effects, where volatility may respond differently to positive and negative shocks.

First approach

```
train <- train %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
train <- na.omit(train) #rimuove prima riga NA
sp_lag <- train$SP_LAG[-1] # allineo
logbr <- diff(log(train$BLACKROCK))[-length(train$BLACKROCK)]
print(length(logbr) == length(sp_lag))
```

```
## [1] TRUE
```

```
garch_spec11s <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd" )
garch_model11s <- ugarchfit(spec = garch_spec11s, data = logbr)

egarch_spec11s <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd")
egarch_model11s <- ugarchfit(spec = egarch_spec11s, data = logbr)

tgarch_spec11s <- ugarchspec(
  variance.model = list(model = "fGARCH", garchOrder = c(2, 1),
                        submodel = "TGARCH"),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd")
tgarch_model11s <- ugarchfit(spec = tgarch_spec11s, data = logbr)

log_diff_test_br <- diff(log(test$BLACKROCK))[-length(test$BLACKROCK)]
test <- test %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
test <- na.omit(test)

predictions_garch11s <- ugarchforecast(garch_model11s, n.ahead = 53,
                        external.forecasts = list(mregfor = test$SP_LAG))
```

```r
rmse_garch11s <- sqrt(mean((predictions_garch11s@forecast$seriesFor
                            - log_diff_test_br)^2))

predictions_egarch11s <- ugarchforecast(egarch_model11s, n.ahead = 53,
                         external.forecasts = list(mregfor = test$SP_LAG))
rmse_egarch11s <- sqrt(mean((predictions_egarch11s@forecast$seriesFor
                             - log_diff_test_br)^2))

predictions_tgarch11s <- ugarchforecast(tgarch_model11s, n.ahead = 53,
                         external.forecasts = list(mregfor = test$SP_LAG))
rmse_tgarch11s <- sqrt(mean((predictions_tgarch11s@forecast$seriesFor
                             - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD:", rmse_garch11s))
```

```
## [1] "RMSE for ARMAX(1,1)-GARCH(2,1)-SSTD: 0.0325767995711624"
```

```r
print(paste("RMSE for ARMAX(1,1)-EGARCH(2,1)-SSTD:", rmse_egarch11s))
```

```
## [1] "RMSE for ARMAX(1,1)-EGARCH(2,1)-SSTD: 0.0325787707730387"
```

```r
print(paste("RMSE for ARMAX(1,1)-TGARCH(2,1)-SSTD:", rmse_tgarch11s))
```

```
## [1] "RMSE for ARMAX(1,1)-TGARCH(2,1)-SSTD: 0.0325980551190459"
```

```r
infocriteria(garch_model11s)*length(train$BLACKROCK)
```

```
##
## Akaike       -1928.796
## Bayes        -1891.098
## Shibata      -1929.124
## Hannan-Quinn -1913.981
```

```r
infocriteria(egarch_model11s)*length(train$BLACKROCK)
```

```
##
## Akaike       -1929.491
## Bayes        -1883.416
## Shibata      -1929.979
## Hannan-Quinn -1911.385
```

```r
infocriteria(tgarch_model11s)*length(train$BLACKROCK)
```

```
##
## Akaike       -1943.621
## Bayes        -1897.546
## Shibata      -1944.108
## Hannan-Quinn -1925.515
```

Given the minimal difference in RMSE between the EGARCH(2,1)-SSTD and TGARCH(2,1)-SSTD models, and considering that the TGARCH specification yields both lower AIC and BIC values, we select the ARMAX(1,1)-TGARCH(2,1)-SSTD model as the final choice.

Second approach

```
train <- data[1:train_size, ]
test <- data[(train_size+1):nrow(data), ]
sp= train$`S&P 500 FIN SVS - PRICE INDEX` [-1]
logbr = diff(log(train$BLACKROCK))
length(sp)==length(logbr)
```

```
## [1] TRUE
```

```
gg22x21sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd" )
garchg22x21sstd <- ugarchfit(spec = gg22x21sstd, data = logbr)

gge22x21sstd <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd")
egarchg22x21sstd <- ugarchfit(spec = gge22x21sstd, data = logbr)

ggt22x21sstd <- ugarchspec(
  variance.model = list(model = "fGARCH", submodel = "TGARCH",
                        garchOrder = c(2,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd")
tgarchg22x21sstd <- ugarchfit(spec = ggt22x21sstd, data = logbr)

sp_test = test$`S&P 500 FIN SVS - PRICE INDEX`[-1]

predictions_garch22x21sstd <- ugarchforecast(garchg22x21sstd,
            n.ahead = 53, external.forecasts = list(mregfor = sp_test))
rmse_garch22x21sstd <-
  sqrt(mean((predictions_garch22x21sstd@forecast$seriesFor
                                  - log_diff_test_br)^2))

predictions_egarch22x21sstd <- ugarchforecast(egarchg22x21sstd,
              n.ahead = 53, external.forecasts = list(mregfor = sp_test))
rmse_egarch22x21sstd <- sqrt(mean((predictions_egarch22x21sstd@forecast$seriesFor
                      - log_diff_test_br)^2))

predictions_tgarch22x21sstd <- ugarchforecast(tgarchg22x21sstd,
            n.ahead = 53, external.forecasts = list(mregfor = sp_test))
rmse_tgarch22x21sstd <-
  sqrt(mean((predictions_tgarch22x21sstd@forecast$seriesFor
             - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(2,2)-GARCH(2,1)-SSTD:", rmse_garch22x21sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-GARCH(2,1)-SSTD: 0.0325058322932511"
```

```r
print(paste("RMSE for ARMAX(2,2)-EGARCH(2,1)-SSTD:", rmse_egarch22x21sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-EGARCH(2,1)-SSTD: 0.0324785966952761"
```

```r
print(paste("RMSE for ARMAX(2,2)-TGARCH(2,1)-SSTD:", rmse_tgarch22x21sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-TGARCH(2,1)-SSTD: 0.0325113110482382"
```

```r
infocriteria(garchg22x21sstd) * length(train$BLACKROCK)
```

```
##
## Akaike       -1934.197
## Bayes        -1888.099
## Shibata      -1934.683
## Hannan-Quinn -1916.083
```

```r
infocriteria(egarchg22x21sstd) * length(train$BLACKROCK)
```

```
##
## Akaike       -1934.311
## Bayes        -1879.832
## Shibata      -1934.987
## Hannan-Quinn -1912.904
```

```r
infocriteria(tgarchg22x21sstd) * length(train$BLACKROCK)
```

```
##
## Akaike       -1947.028
## Bayes        -1892.549
## Shibata      -1947.704
## Hannan-Quinn -1925.621
```

Although the TGARCH(2,1)-SSTD model presents slightly better values for AIC and BIC, the EGARCH(2,1)-SSTD model achieves the lowest RMSE among all specifications. Given the focus on predictive accuracy, the EGARCH model is selected.

```r
egarchg22x21sstd
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(2,1)
## Mean Model   : ARFIMA(2,0,2)
```

```
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1     1.131932    0.081504  13.88810 0.000000
## ar2    -0.319175    0.048159  -6.62753 0.000000
## ma1    -1.017885    0.078121 -13.02960 0.000000
## ma2     0.171492    0.040761   4.20731 0.000026
## mxreg1  0.000002    0.000002   1.43472 0.151368
## omega  -0.970154    0.418289  -2.31934 0.020376
## alpha1 -0.112815    0.081333  -1.38708 0.165417
## alpha2 -0.046610    0.089957  -0.51814 0.604362
## beta1   0.857848    0.060934  14.07822 0.000000
## gamma1  0.016071    0.123973   0.12963 0.896857
## gamma2  0.158151    0.127317   1.24218 0.214169
## skew    0.849805    0.056009  15.17272 0.000000
## shape   8.384040    3.248588   2.58083 0.009856
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1     1.131932    0.081726  13.85033 0.000000
## ar2    -0.319175    0.018529 -17.22588 0.000000
## ma1    -1.017885    0.075486 -13.48438 0.000000
## ma2     0.171492    0.021052   8.14612 0.000000
## mxreg1  0.000002    0.000002   1.19845 0.230741
## omega  -0.970154    0.441674  -2.19654 0.028053
## alpha1 -0.112815    0.077644  -1.45297 0.146231
## alpha2 -0.046610    0.090954  -0.51246 0.608328
## beta1   0.857848    0.065164  13.16441 0.000000
## gamma1  0.016071    0.140579   0.11432 0.908984
## gamma2  0.158151    0.154442   1.02402 0.305827
## skew    0.849805    0.048400  17.55799 0.000000
## shape   8.384040    3.037453   2.76022 0.005776
##
## LogLikelihood : 978.1616
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.9883
## Bayes        -3.8759
## Shibata      -3.9897
## Hannan-Quinn -3.9441
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2878  0.5916
## Lag[2*(p+q)+(p+q)-1][11]    3.1575  1.0000
## Lag[4*(p+q)+(p+q)-1][19]    5.3274  0.9895
## d.o.f=4
## H0 : No serial correlation
##
```

```
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.04593 0.83030
## Lag[2*(p+q)+(p+q)-1][8]   10.06637 0.03358
## Lag[4*(p+q)+(p+q)-1][14]  12.98750 0.06120
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]    0.8751 0.500 2.000  0.3496
## ARCH Lag[6]    0.8775 1.461 1.711  0.7832
## ARCH Lag[8]    1.4096 2.368 1.583  0.8563
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.2701
## Individual Statistics:
## ar1    0.04031
## ar2    0.03389
## ma1    0.05803
## ma2    0.04604
## mxreg1 0.12573
## omega  0.25108
## alpha1 0.04476
## alpha2 0.05901
## beta1  0.23572
## gamma1 0.04378
## gamma2 0.05229
## skew   0.37005
## shape  0.31858
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.89 3.15 3.69
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.3412 0.1805
## Negative Sign Bias  0.7097 0.4782
## Positive Sign Bias  0.4289 0.6682
## Joint Effect        1.9846 0.5756
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     18.31       0.5016
## 2    30     24.88       0.6842
## 3    40     37.16       0.5542
## 4    50     45.75       0.6056
##
```

```
##
## Elapsed time : 0.7442441
```

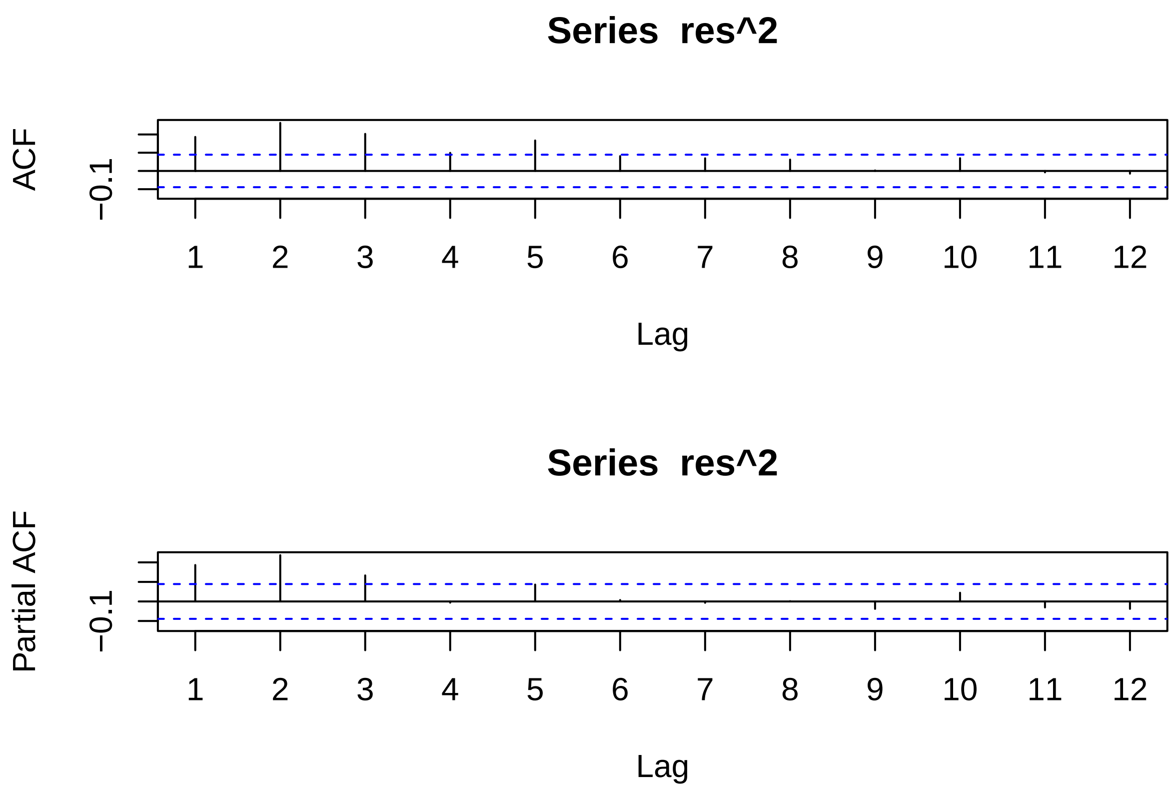
Looking at the model output, the squared residuals still show significant autocorrelation, suggesting that the model does not fully capture the heteroskedasticity and needs to be revised.

We check for autocorrelation in the residuals and in the squared residuals.

```
res = egarchg22x21sstd@fit$residuals
Box.test(res, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 11.278, df = 12, p-value = 0.5052
```

```
par(mfrow=c(2,1))
Acf(res^2 , lag=12)
Pacf(res^2, lag=12)
```




```
par(mfrow=c(1,1))
Box.test(res^2, lag = 12, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 99.665, df = 12, p-value = 6.661e-16
```

The Ljung-Box test on the residuals does not indicate any issues, suggesting that the ARMA part of the model is well specified. The ACF and PACF plots of the squared residuals show significant lags, and the Ljung-Box test on squared residuals rejects the null hypothesis. This indicates the presence of remaining autocorrelation in the variance, suggesting that the GARCH model may be misspecified.

```
gge22x31sstd <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(3,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd"
)

egarchg22x31sstd <- ugarchfit(spec = gge22x31sstd, data = logbr)

predictions_egarch22x31sstd <- ugarchforecast(egarchg22x31sstd,
            n.ahead = 53, external.forecasts = list(mregfor = sp_test))
rmse_egarch22x31sstd <- sqrt(mean((predictions_egarch22x31sstd@forecast$seriesFor
                                   - log_diff_test_br)^2))

print(paste("RMSE for ARMAX(2,2)-EGARCH(2,1)-SSTD:", rmse_egarch22x21sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-EGARCH(2,1)-SSTD: 0.0324785966952761"
```

```
print(paste("RMSE for ARMAX(2,2)-EGARCH(3,1)-SSTD:", rmse_egarch22x31sstd))
```

```
## [1] "RMSE for ARMAX(2,2)-EGARCH(3,1)-SSTD: 0.0324758683722274"
```

```
infocriteria(egarchg22x21sstd) * length(train$BLACKROCK)
```

```
##
## Akaike       -1934.311
## Bayes        -1879.832
## Shibata      -1934.987
## Hannan-Quinn -1912.904
```

```
infocriteria(egarchg22x31sstd) * length(train$BLACKROCK)
```

```
##
## Akaike       -1931.834
## Bayes        -1868.973
## Shibata      -1932.729
## Hannan-Quinn -1907.133
```

After modifying the model to ARMAX(2,2)-EGARCH(3,1)-SSTD, the AIC and BIC values slightly worsen, but the RMSE improves marginally. More importantly, the residual diagnostics show no significant issues at the 5% level, suggesting that the model has successfully addressed previous autocorrelation problems. Based on these results, the model ARMAX(2,2)-GARCH(3,1)-SSTD is selected as the final specification.

Forecast

Now, we implement two approaches for forecasting.

```r
# First approach
data = read_excel(here('blackrock.xlsx'), sheet = 6, col_names = T)
logbr = diff(log(data$BLACKROCK))
last_logbr <- tail(log(data$BLACKROCK),1)
last_sp_lag <- tail(data$`S&P 500 FIN SVS - PRICE INDEX`, 1)
data <- data %>%
  mutate(SP_LAG = lag(`S&P 500 FIN SVS - PRICE INDEX`, 1))
sp_lag <- data$SP_LAG[-1]

tgarch_spec11s <- ugarchspec(
  variance.model = list(model = "fGARCH", garchOrder = c(2, 1),
                        submodel = "TGARCH"),
  mean.model = list(armaOrder = c(1, 1), include.mean = FALSE,
                    external.regressors = as.matrix(sp_lag)),
  distribution.model = "sstd")
tgarch_model11s <- ugarchfit(spec = tgarch_spec11s, data = logbr)

forecast <- ugarchforecast(tgarch_model11s, n.ahead = 1,
                  external.forecasts = list(xregfor =last_sp_lag))
predicted_log_returns <- forecast@forecast$seriesFor
predicted_price <- exp(last_logbr + predicted_log_returns)
cat("Prediction of the price for the next week (first approach): ", predicted_price, "\n")
```

```
## Prediction of the price for the next week (first approach):  894.7273
```

```r
# second approach
data = read_excel(here('blackrock.xlsx'), sheet = 6, col_names = T)
sp= data$`S&P 500 FIN SVS - PRICE INDEX` [-1]
logbr = diff(log(data$BLACKROCK))

gge22x31sstd <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(3,1)),
  mean.model = list(armaOrder = c(2,2), include.mean = FALSE,
                    external.regressors = as.matrix(sp)),
  distribution.model = "sstd")
egarchg22x21sstd <- ugarchfit(spec = gge22x31sstd, data = logbr)

last_log_sp <- log(tail(data$`S&P 500 FIN SVS - PRICE INDEX`, 1))
mod3 =  garchFit(diff(log(data$`S&P 500 FIN SVS - PRICE INDEX`)) ~
                   arma(4,2) + garch(3,0),
                 data = diff(log(data$`S&P 500 FIN SVS - PRICE INDEX`)) ,
                 trace = F, include.mean = F, cond.dist = "sstd")
forsp = predict(mod3, n.ahead = 1)
predicted_log_sp <- last_log_sp + forsp$meanForecast
predicted_sp <- exp(predicted_log_sp)
forecast2 <- ugarchforecast(egarchg22x31sstd, n.ahead = 1,
                    external.forecasts = list(xregfor =predicted_sp))
predicted_log_returns2 <- forecast2@forecast$seriesFor
predicted_price2 <- exp(last_logbr + predicted_log_returns2)
cat("Prediction of the price for the next week (second approach):", predicted_price2, "\n")
```

```
## Prediction of the price for the next week (second approach): 887.9654
```

```r
cat("Last observed value of external variable (used in the first
    approach with lag): 1372.7")
```

```
## Last observed value of external variable (used in the first
##     approach with lag): 1372.7
```

```r
cat("Predicted value of external variable (used in the second
    approach with exogenous variable aligned):", predicted_sp, "\n")
```

```
## Predicted value of external variable (used in the second
##     approach with exogenous variable aligned): 1371.781
```

The forecasted price for the next week using the first approach (with lagged external variable) is 894.73. The model used here was ARMAX(1,1)-TGARCH(2,1)-SSTD.
The forecasted price using the second approach (with predicted external variable) is 899.5872. The model used here was ARMAX(2,2)-EGARCH(3,1)-SSTD.
The last observed value of the external variable was 1372.7, while the predicted value (ARMA(4,2)-GARCH(3,0)-SSTD) used in the second approach was 1371.78.

```r
predicted_price1 <- 894.7273
predicted_price2 <- 899.5872
real_price <- 914.26

abs_error1 <- abs(real_price - predicted_price1)
abs_error2 <- abs(real_price - predicted_price2)

perc_error1 <- (abs_error1 / real_price) * 100
perc_error2 <- (abs_error2 / real_price) * 100

cat("Absolute error (first approach):", abs_error1, "\n")
```

```
## Absolute error (first approach): 19.5327
```

```r
cat("Percentage error (first approach):", perc_error1, "%\n\n")
```

```
## Percentage error (first approach): 2.136449 %
```

```r
cat("Absolute error (second approach):", abs_error2, "\n")
```

```
## Absolute error (second approach): 14.6728
```

```r
cat("Percentage error (second approach):", perc_error2, "%\n")
```

```
## Percentage error (second approach): 1.604883 %
```

WEEK 7

Introduction

This week, we focus on modeling and forecasting the returns of BlackRock using a Vector Autoregression (VAR) model, which allows us to analyze multiple time series simultaneously while accounting for their dynamic interactions. In addition to the price series of BlackRock, we include two other variables: the S&P 500 index, representing the overall performance of the U.S. stock market, and the VIX index, commonly known as the "fear index," which measures expected market volatility and reflects the level of uncertainty in financial markets. These two variables are included to capture macroeconomic and market-wide influences that may affect the behavior of an individual stock like BlackRock, thereby enhancing the quality of the forecasts compared to a univariate model.

```r
data <- read_excel(here("blackrock.xlsx"), sheet = 7, col_names = TRUE)
colnames(data) <- c("Date", "br", "sp", "vi")
```

Here, we test the stationarity of the variables using the Augmented Dickey-Fuller (ADF) test and apply differencing to non-stationary series.

```r
adf.test(data$br)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$br
## Dickey-Fuller = -2.6575, Lag order = 8, p-value = 0.3
## alternative hypothesis: stationary
```

```r
adf.test(data$sp)
```

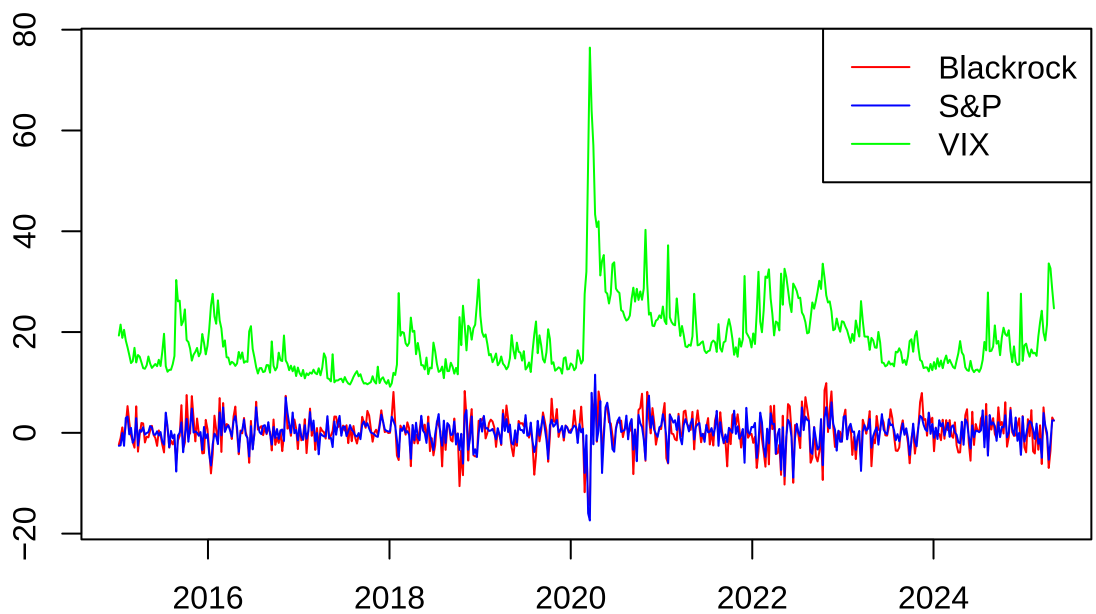
```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$sp
## Dickey-Fuller = -2.2287, Lag order = 8, p-value = 0.4815
## alternative hypothesis: stationary
```

```r
adf.test(data$vi)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$vi
## Dickey-Fuller = -4.3296, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
data$br = c(NA,100*diff(log(data$br)))
data$sp = c(NA,100*diff(log(data$sp)))
data = data[-1,]

train_size <- floor(0.9 * nrow(data))
train <- data[1:train_size, ]
test <- data[(train_size + 1):nrow(data), ]

br <- train$br
sp <- train$sp

ylim_range <- range(c(data$br, data$sp, data$vi), na.rm = TRUE)
plot(data$Date, data$br, type = "l", col = "red", xlab = " ",
     ylab = " ", ylim = ylim_range)
lines(data$Date, data$sp, col = "blue")
lines(data$Date, data$vi, col = "green")
legend("topright", legend = c("Blackrock", "S&P", "VIX"),
       col = c("red", "blue", "green"), lty = 1)
```



In this section, we select the lag order for the VAR model. The function suggests a lag order of 1, and we proceed to estimate the model with that lag order.

VAR model

```
VARselect(train[c("br","sp","vi")], lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                 1          2          3          4          5          6
## AIC(n)   4.925193   4.940199   4.953278   4.972142   4.975254   4.996185
## HQ(n)    4.966555   5.012582   5.056682   5.106568   5.140700   5.192653
## SC(n)    5.030372   5.124261   5.216224   5.313972   5.395968   5.495783
## FPE(n) 137.716104 139.798929 141.641172 144.341706 144.796636 147.867047
##                 7          8          9         10
## AIC(n)   4.996158   4.971791   4.992979   5.013707
## HQ(n)    5.223647   5.230301   5.282510   5.334259
## SC(n)    5.574640   5.629157   5.729228   5.828840
## FPE(n) 147.873498 144.327280 147.435257 150.545314
```

```
var1 <- vars::VAR(train[c("br","sp","vi")], p = 1)
summary(var1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: br, sp, vi
## Deterministic variables: const
## Sample size: 484
## Log Likelihood: -3234.156
## Roots of the characteristic polynomial:
## 0.8717 0.1459 0.01891
## Call:
## vars::VAR(y = train[c("br", "sp", "vi")], p = 1)
##
##
## Estimation results for equation br:
## ===================================
## br = br.l1 + sp.l1 + vi.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## br.l1  0.25023    0.08193   3.054  0.00238 **
## sp.l1 -0.15359    0.10741  -1.430  0.15338
## vi.l1  0.06238    0.02246   2.777  0.00569 **
## const -1.00117    0.44403  -2.255  0.02460 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.431 on 480 degrees of freedom
## Multiple R-Squared: 0.03945, Adjusted R-squared: 0.03344
## F-statistic:  6.57 on 3 and 480 DF,  p-value: 0.0002328
```

```
##
##
## Estimation results for equation sp:
## ===================================
## sp = br.l1 + sp.l1 + vi.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## br.l1  0.19723    0.06445   3.060  0.00234 **
## sp.l1 -0.13342    0.08450  -1.579  0.11501
## vi.l1  0.02927    0.01767   1.657  0.09827 .
## const -0.37811    0.34930  -1.082  0.27959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 2.699 on 480 degrees of freedom
## Multiple R-Squared: 0.02912, Adjusted R-squared: 0.02306
## F-statistic:   4.8 on 3 and 480 DF,  p-value: 0.002643
##
##
## Estimation results for equation vi:
## ===================================
## vi = br.l1 + sp.l1 + vi.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## br.l1 -0.24817    0.08341  -2.975  0.00307 **
## sp.l1  0.25532    0.10935   2.335  0.01997 *
## vi.l1  0.88183    0.02287  38.564  < 2e-16 ***
## const  2.16475    0.45206   4.789 2.24e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.493 on 480 degrees of freedom
## Multiple R-Squared: 0.7749,  Adjusted R-squared: 0.7735
## F-statistic: 550.8 on 3 and 480 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##        br     sp     vi
## br 11.769  7.701 -7.945
## sp  7.701  7.283 -7.249
## vi -7.945 -7.249 12.198
##
## Correlation matrix of residuals:
##         br      sp      vi
## br  1.0000  0.8318 -0.6631
## sp  0.8318  1.0000 -0.7691
## vi -0.6631 -0.7691  1.0000
```

We perform Granger causality tests to check the relationships between variables.

```r
grangertest(data$br ~ data$sp, data = train, order=1)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:1) + Lags(data$sp, 1:1)
## Model 2: data$br ~ Lags(data$br, 1:1)
##   Res.Df Df      F  Pr(>F)
## 1    535
## 2    536 -1 4.2477 0.03979 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(data$br ~ data$vi, data = train, order=1)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:1) + Lags(data$vi, 1:1)
## Model 2: data$br ~ Lags(data$br, 1:1)
##   Res.Df Df      F   Pr(>F)
## 1    535
## 2    536 -1 9.9294 0.001718 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(data$br ~ data$sp, data = train, order=2)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:2) + Lags(data$sp, 1:2)
## Model 2: data$br ~ Lags(data$br, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    532
## 2    534 -2 2.2548 0.1059
```

```r
grangertest(data$br ~ data$vi, data = train, order=2)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:2) + Lags(data$vi, 1:2)
## Model 2: data$br ~ Lags(data$br, 1:2)
##   Res.Df Df      F   Pr(>F)
## 1    532
## 2    534 -2 5.8046 0.003208 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(data$br ~ data$sp, data = train, order=3)
```

```
## Granger causality test
##
```

```
## Model 1: data$br ~ Lags(data$br, 1:3) + Lags(data$sp, 1:3)
## Model 2: data$br ~ Lags(data$br, 1:3)
##   Res.Df Df      F  Pr(>F)
## 1    529
## 2    532 -3 2.7148 0.04419 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data$br ~ data$vi, data = train, order=3)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:3) + Lags(data$vi, 1:3)
## Model 2: data$br ~ Lags(data$br, 1:3)
##   Res.Df Df      F    Pr(>F)
## 1    529
## 2    532 -3 5.7547 0.0007053 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
causality(var1,cause = c("br"))$Granger
```

```
##
##  Granger causality H0: br do not Granger-cause sp vi
##
## data:  VAR object var1
## F-Test = 5.1557, df1 = 2, df2 = 1440, p-value = 0.005873
```

```
causality(var1,cause = c("sp"))$Granger
```

```
##
##  Granger causality H0: sp do not Granger-cause br vi
##
## data:  VAR object var1
## F-Test = 2.7381, df1 = 2, df2 = 1440, p-value = 0.06503
```

```
causality(var1,cause = c("vi"))$Granger
```

```
##
##  Granger causality H0: vi do not Granger-cause br sp
##
## data:  VAR object var1
## F-Test = 4.5501, df1 = 2, df2 = 1440, p-value = 0.01072
```

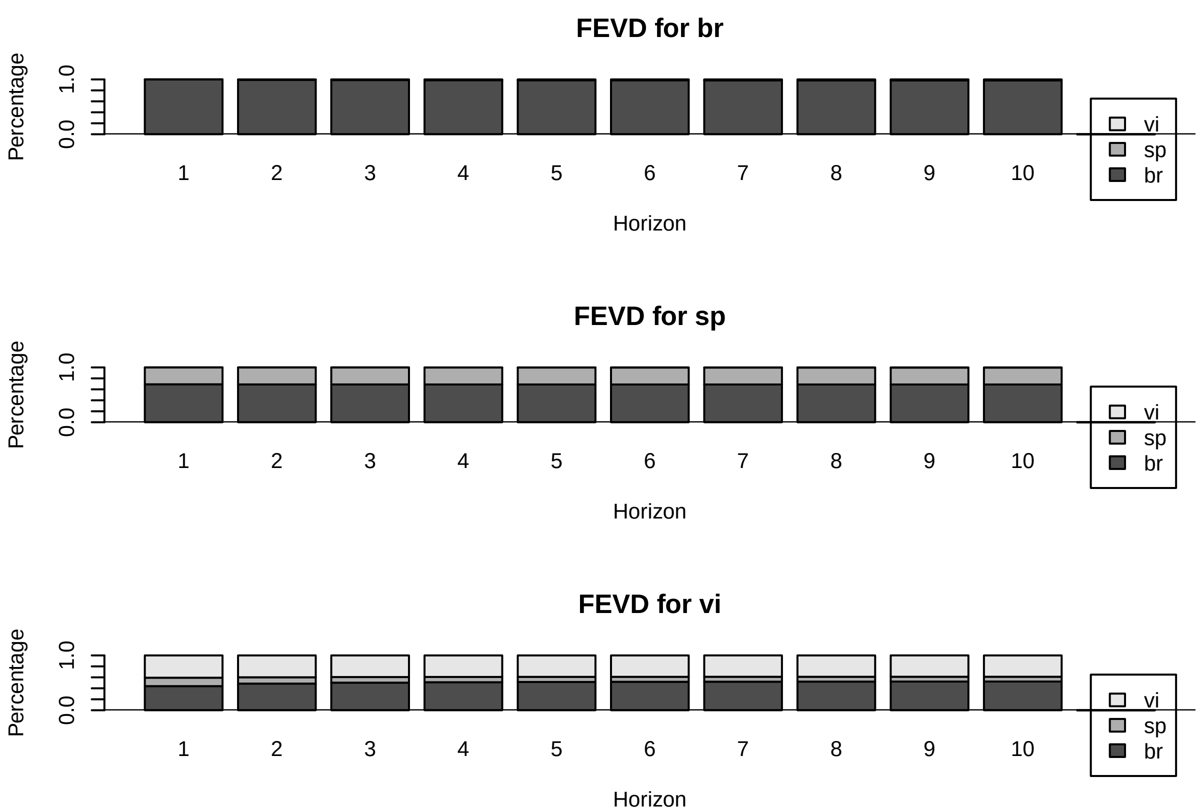
```
causality(var1,cause = c("vi","sp"))$Granger
```

```
##
##  Granger causality H0: sp vi do not Granger-cause br
##
## data:  VAR object var1
## F-Test = 6.2028, df1 = 2, df2 = 1440, p-value = 0.002078
```

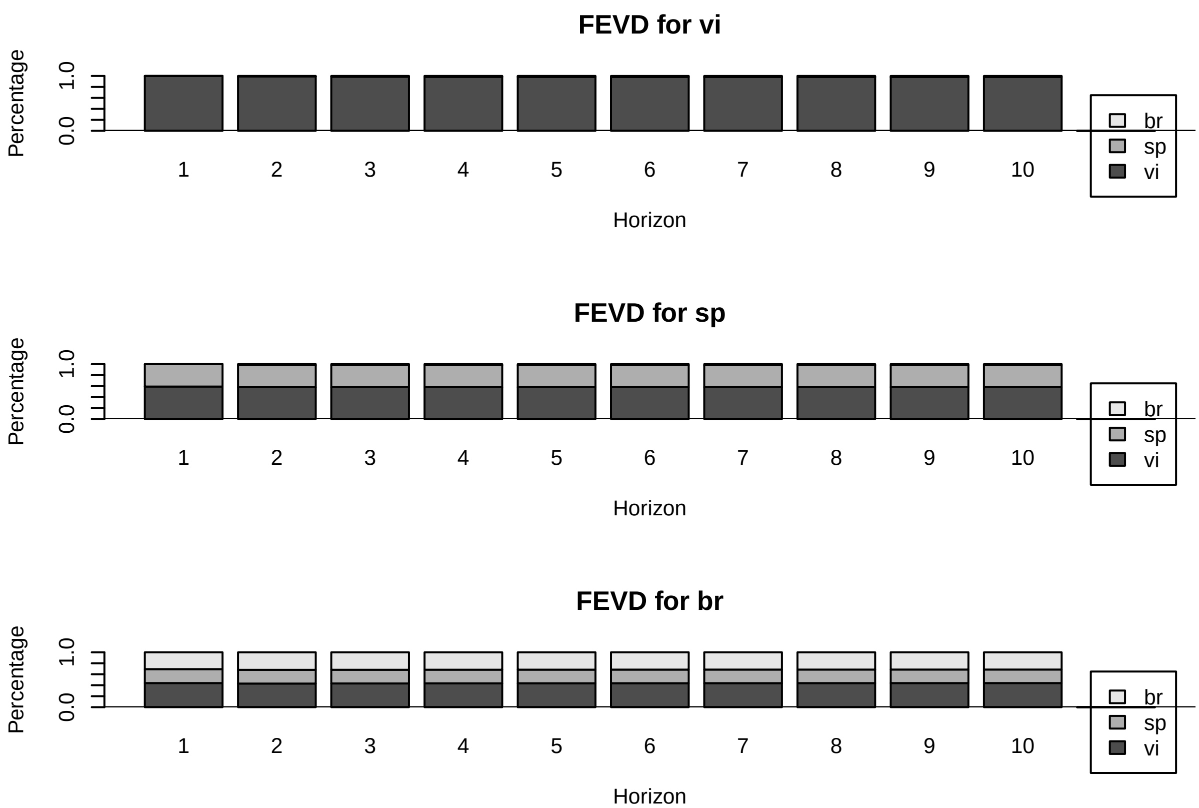
So, from the Granger causality tests, we see that both `sp` (S&P 500) and `vi` (VIX) have an effect on `br` (BlackRock), especially at lag 1. `vi` is pretty strong, while `sp` is a bit weaker. When we check lag 2, `vi` still has a clear influence, but `sp` doesn't really seem to matter anymore. At lag 3, both `sp` (S&P 500) and `vi` (VIX) are back to having a significant effect on `br` (BlackRock).

This section analyzes the impulse response functions (IRF) and variance decomposition (FEVD) to understand the dynamic relationships between the variables. Additionally we reverse the order of the variables in the VAR model to check the robustness of the results and see if the ordering affects the variance decomposition. This helps ensure that our conclusions aren't just driven by the way we ordered the variables.

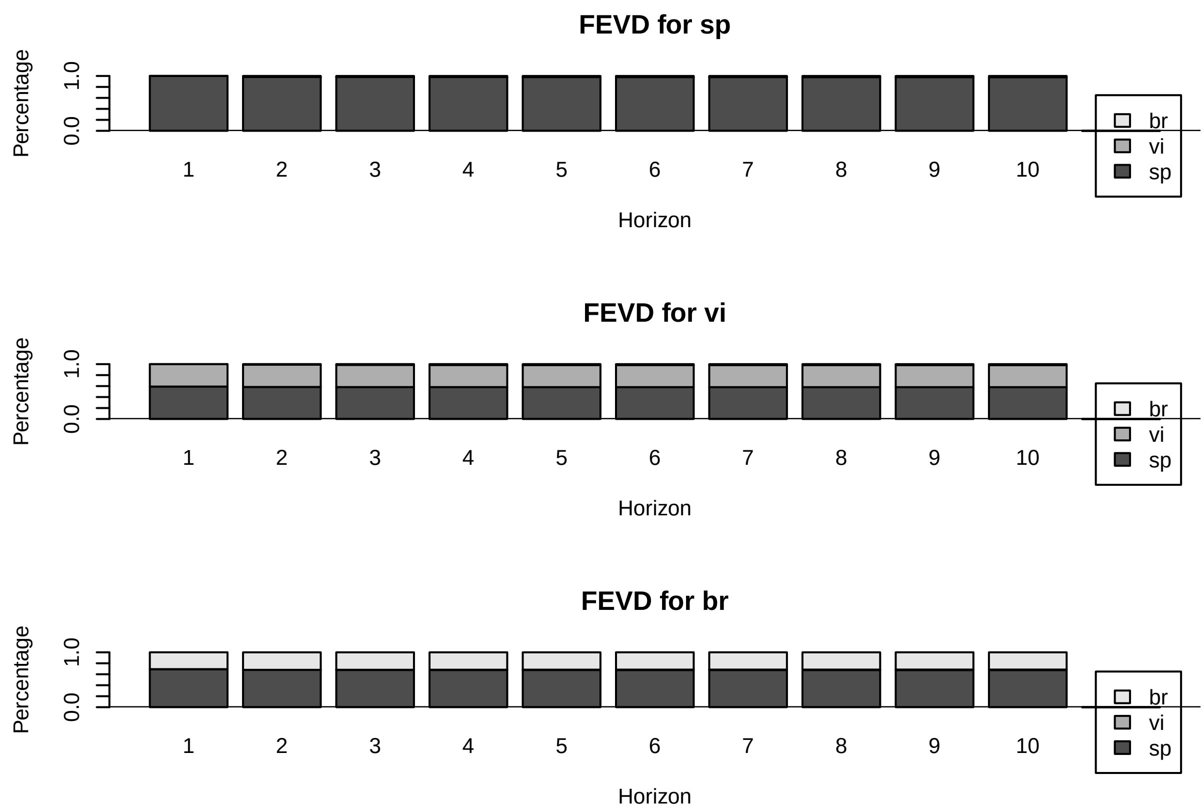
```
ir = irf(var1,n.ahead = 10)

vd = fevd(var1,n.ahead = 10)
plot(vd)
```




```
var1_reverse1 <- vars::VAR(train[c("vi","sp","br")],p = 1)
vd_reverse1 = fevd(var1_reverse1 , n.ahead = 10)
plot(vd_reverse1)
```




```
var1_reverse2 <- vars::VAR(train[c("sp","vi","br")],p = 1)
vd_reverse2 = fevd(var1_reverse2,n.ahead = 10)
plot(vd_reverse2)
```

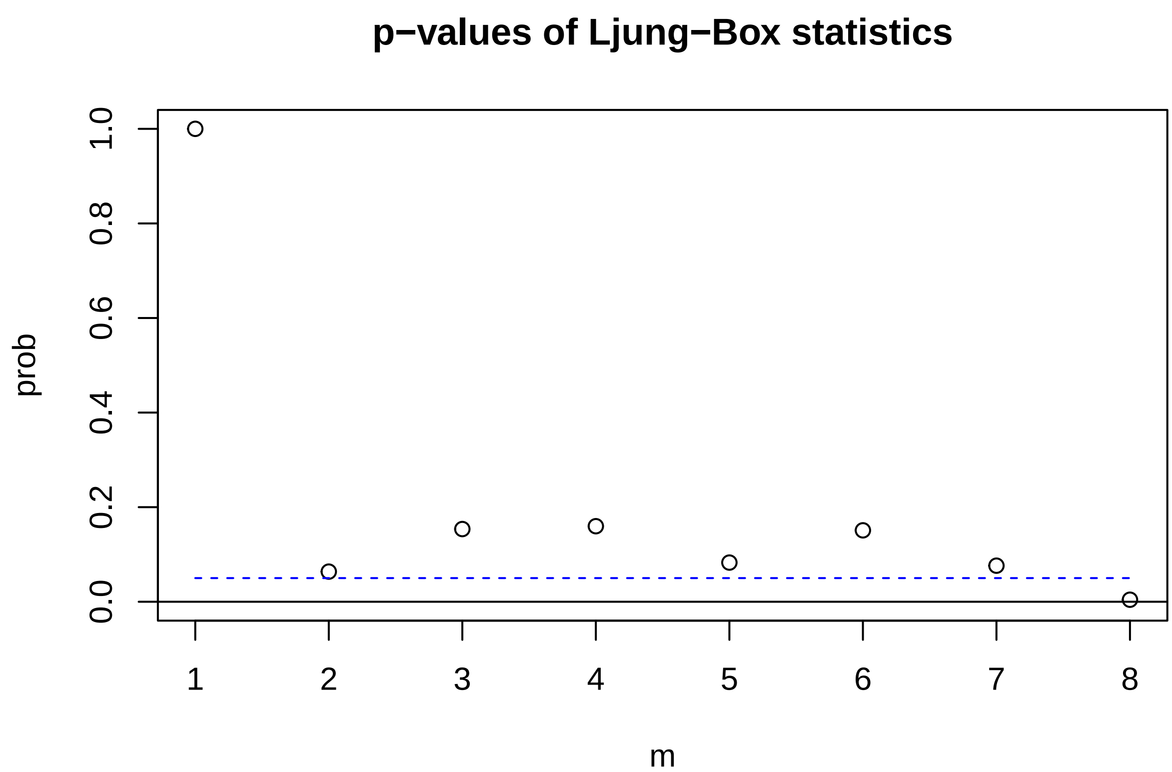


The variance decomposition (FEVD) analysis revealed that the results are sensitive to the ordering of variables in the VAR model, suggesting that the causal structure implied by the ordering can influence the economic interpretation of the results.

To ensure consistency with sound economic reasoning, we chose the ordering VIX → S&P 500 → BlackRock. This choice reflects the assumption that the VIX, as a measure of market uncertainty, is the most exogenous variable; the S&P 500, representing the overall stock market, is influenced by uncertainty but also affects individual stocks; and BlackRock, as a single stock, is the most endogenous variable, responding to both market-wide movements and changes in uncertainty.

```r
res = cbind(var1_reverse1$varresult$br$residuals,
            var1_reverse1$varresult$sp$residuals,
            var1_reverse1$varresult$vi$residuals)
mq(res, adj=9, lag = 8)
```

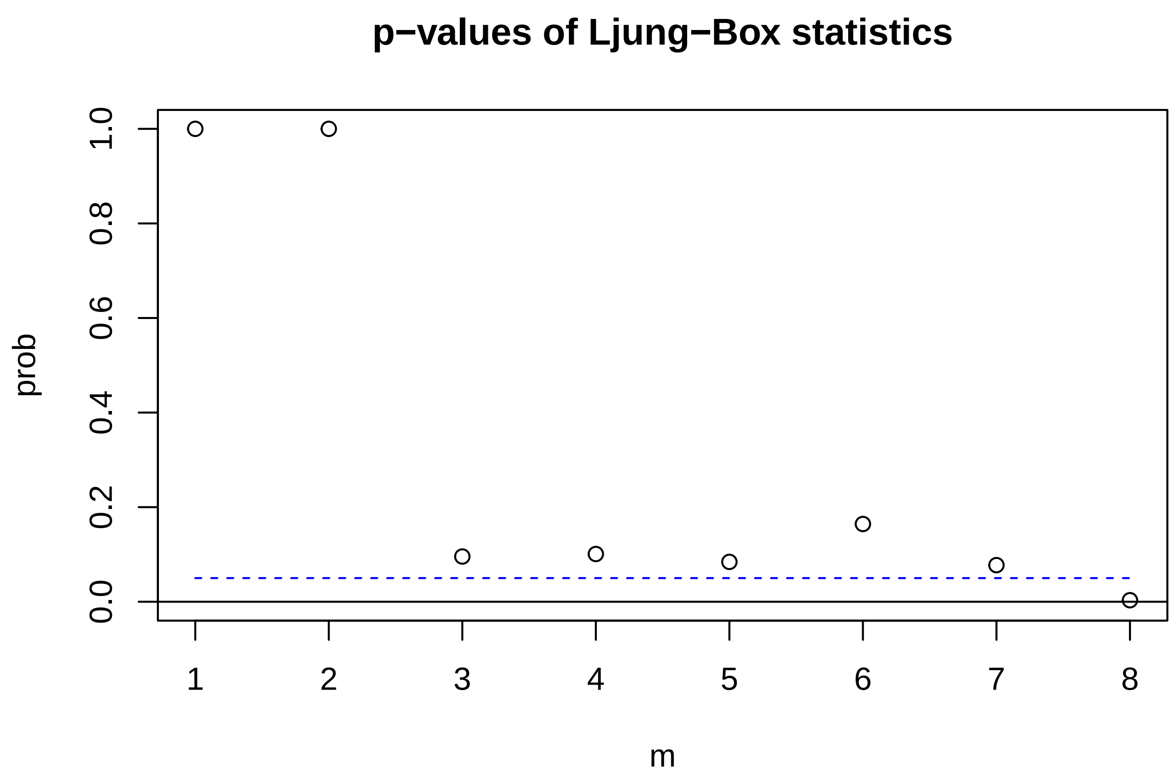
```
## Ljung-Box Statistics:
##         m       Q(m)     df    p-value
## [1,]   1.00      5.28    0.00     1.00
## [2,]   2.00     16.16    9.00     0.06
## [3,]   3.00     24.04   18.00     0.15
## [4,]   4.00     34.22   27.00     0.16
## [5,]   5.00     48.29   36.00     0.08
## [6,]   6.00     54.76   45.00     0.15
## [7,]   7.00     69.49   54.00     0.08
## [8,]   8.00     96.42   63.00     0.00
```



The Ljung-Box test shows that up to lag 7, the p-values are all above the 5% threshold, indicating no significant residual autocorrelation.

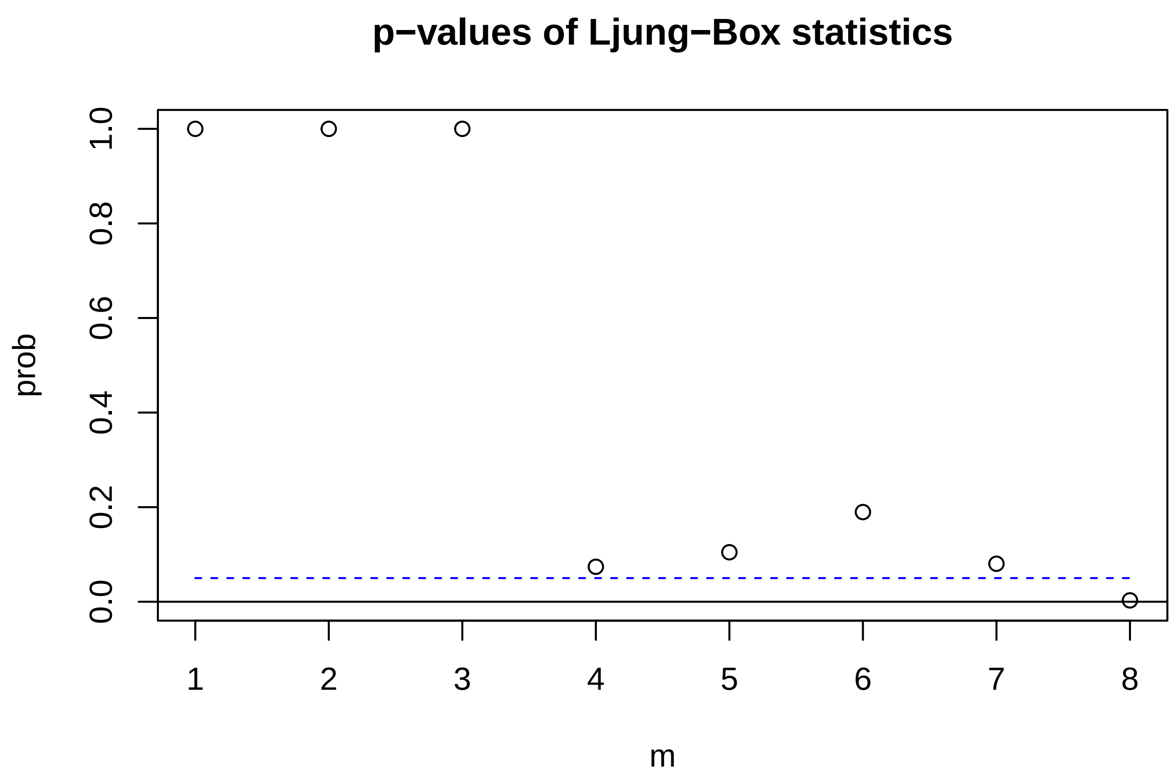
```
var2 <- vars::VAR(train[c("vi","sp","br")], p = 2)
res2 <- cbind(var2$varresult$br$residuals,
              var2$varresult$sp$residuals,
              var2$varresult$vi$residuals)
mq(res2, lag = 8, adj = 18)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
## [1,]  1.000     0.155  -9.000     1.00
## [2,]  2.000     8.555   0.000     1.00
## [3,]  3.000    14.834   9.000     0.10
## [4,]  4.000    25.949  18.000     0.10
## [5,]  5.000    37.607  27.000     0.08
## [6,]  6.000    44.178  36.000     0.16
## [7,]  7.000    59.095  45.000     0.08
## [8,]  8.000    86.607  54.000     0.00
```




```r
var3 <- vars::VAR(train[c("vi","sp","br")], p = 3)
res3 <- cbind(var3$varresult$br$residuals,
              var3$varresult$sp$residuals,
              var3$varresult$vi$residuals)
mq(res3, lag = 8, adj = 27)
```

```
## Ljung-Box Statistics:
##           m       Q(m)      df    p-value
## [1,]   1.0000     0.0996 -18.0000     1.00
## [2,]   2.0000     0.3845  -9.0000     1.00
## [3,]   3.0000     3.1554   0.0000     1.00
## [4,]   4.0000    15.6775   9.0000     0.07
## [5,]   5.0000    25.7899  18.0000     0.10
## [6,]   6.0000    33.2280  27.0000     0.19
## [7,]   7.0000    48.4548  36.0000     0.08
## [8,]   8.0000    75.6079  45.0000     0.00
```




```
cbind(AIC(var1_reverse1),AIC(var2), AIC(var3))
```

```
##          [,1]     [,2]     [,3]
## [1,] 6492.312 6487.456 6481.886
```

```
cbind(BIC(var1_reverse1),BIC(var2), BIC(var3))
```

```
##          [,1]     [,2]     [,3]
## [1,] 6542.497 6575.237 6607.224
```

```
pred1 <- predict(var1_reverse1, n.ahead = nrow(test))
pred2 <- predict(var2, n.ahead = nrow(test))
pred3 <- predict(var3, n.ahead = nrow(test))

br_pred1 <- pred1$fcst$br[, "fcst"]
br_pred2 <- pred2$fcst$br[, "fcst"]
br_pred3 <- pred3$fcst$br[, "fcst"]

rmse1_br <- sqrt(mean((test$br - br_pred1)^2, na.rm = TRUE))
rmse2_br <- sqrt(mean((test$br - br_pred2)^2, na.rm = TRUE))
rmse3_br <- sqrt(mean((test$br - br_pred3)^2, na.rm = TRUE))

cbind(rmse1_br, rmse2_br, rmse3_br)
```

```
##      rmse1_br rmse2_br rmse3_br
## [1,] 3.239006 3.240471  3.24164
```

We choose VAR with p=1 because it has the lowest RMSE and also the smallest BIC.

Forecast

Now, we proceed with forecasting for the next week using the model we have estimated.

```r
rawdata = read_excel(here("blackrock.xlsx"), sheet = 7, col_names = TRUE)
colnames(rawdata) <- c( "Date", "br", "sp","vi")

last <- tail(rawdata$br, 1)
log_last <- log(last)

var1 <- vars::VAR(data[c("vi", "sp", "br")], p = 1)
for1 <- predict(var1, n.ahead = 1, ci = 0.95)
br_logret1 <- for1$fcst$br[1]

log_forecast <- log_last + (br_logret1 / 100)

forecast_price <- exp(log_forecast)
print(forecast_price)
```

```
## [1] 920.7407
```

```r
predicted_price1 <- 920.7407
real_price <- 920.3601

abs_error1 <- abs(real_price - predicted_price1)
perc_error1 <- (abs_error1 / real_price) * 100

cat("Absolute error:", abs_error1, "\n")
```

```
## Absolute error: 0.3806
```

```r
cat("Percentage error:", perc_error1, "%\n\n")
```

```
## Percentage error: 0.04135338 %
```

WEEK 8

Introduction

We use the same VAR model previously estimated, we update the data to include the most recent observations and generate a new forecast for the next period.

```r
data <- read_excel(here("blackrock.xlsx"), sheet = 8, col_names = TRUE)
colnames(data) <- c("Date", "br", "sp", "vi")
```

```r
adf.test(data$br)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$br
## Dickey-Fuller = -2.6634, Lag order = 8, p-value = 0.2975
## alternative hypothesis: stationary
```

```r
adf.test(data$sp)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$sp
## Dickey-Fuller = -2.1876, Lag order = 8, p-value = 0.4989
## alternative hypothesis: stationary
```

```r
adf.test(data$vi)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$vi
## Dickey-Fuller = -4.338, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
data$br <- c(NA, 100 * diff(log(data$br)))
data$sp <- c(NA, 100 * diff(log(data$sp)))
data <- data[-1,]

train_size <- floor(0.9 * nrow(data))
train <- data[1:train_size, ]
test <- data[(train_size + 1):nrow(data), ]
```

```r
VARselect(train[c("vi","sp","br")], lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                1          2          3          4          5          6
## AIC(n)  4.921179   4.936001   4.948932   4.967441   4.970478   4.991424
## HQ(n)   4.962471   5.008262   5.052162   5.101640   5.135646   5.187561
```

```
## SC(n)    5.026190   5.119770   5.211458   5.308725   5.390520   5.490224
## FPE(n) 137.164423 139.213323 141.026895 143.664629 144.106721 147.164580
##                 7          8          9         10
## AIC(n)    4.991726   4.967295   4.988653   5.009549
## HQ(n)     5.218831   5.225369   5.277696   5.329561
## SC(n)     5.569284   5.623611   5.723726   5.823381
## FPE(n) 147.219330 143.679496 146.798469 149.920138
```

```r
var1 <- vars::VAR(train[c("vi","sp","br")], p = 1)
```

```r
grangertest(data$br ~ data$sp, data = train, order = 1)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:1) + Lags(data$sp, 1:1)
## Model 2: data$br ~ Lags(data$br, 1:1)
##   Res.Df Df      F  Pr(>F)
## 1    536
## 2    537 -1 4.2446 0.03986 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(data$br ~ data$vi, data = train, order = 1)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:1) + Lags(data$vi, 1:1)
## Model 2: data$br ~ Lags(data$br, 1:1)
##   Res.Df Df      F   Pr(>F)
## 1    536
## 2    537 -1 9.9513 0.001698 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(data$br ~ data$sp, data = train, order = 2)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:2) + Lags(data$sp, 1:2)
## Model 2: data$br ~ Lags(data$br, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    533
## 2    535 -2 2.2546 0.1059
```

```r
grangertest(data$br ~ data$vi, data = train, order = 2)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:2) + Lags(data$vi, 1:2)
## Model 2: data$br ~ Lags(data$br, 1:2)
```

```
##   Res.Df Df      F   Pr(>F)
## 1    533
## 2    535 -2 5.8191 0.003162 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data$br ~ data$sp, data = train, order = 3)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:3) + Lags(data$sp, 1:3)
## Model 2: data$br ~ Lags(data$br, 1:3)
##   Res.Df Df     F  Pr(>F)
## 1    530
## 2    533 -3 2.715 0.04418 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data$br ~ data$vi, data = train, order = 3)
```

```
## Granger causality test
##
## Model 1: data$br ~ Lags(data$br, 1:3) + Lags(data$vi, 1:3)
## Model 2: data$br ~ Lags(data$br, 1:3)
##   Res.Df Df      F    Pr(>F)
## 1    530
## 2    533 -3 5.7615 0.0006986 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
causality(var1, cause = c("br"))$Granger
```

```
##
##  Granger causality H0: br do not Granger-cause vi sp
##
## data:  VAR object var1
## F-Test = 5.0524, df1 = 2, df2 = 1443, p-value = 0.006508
```

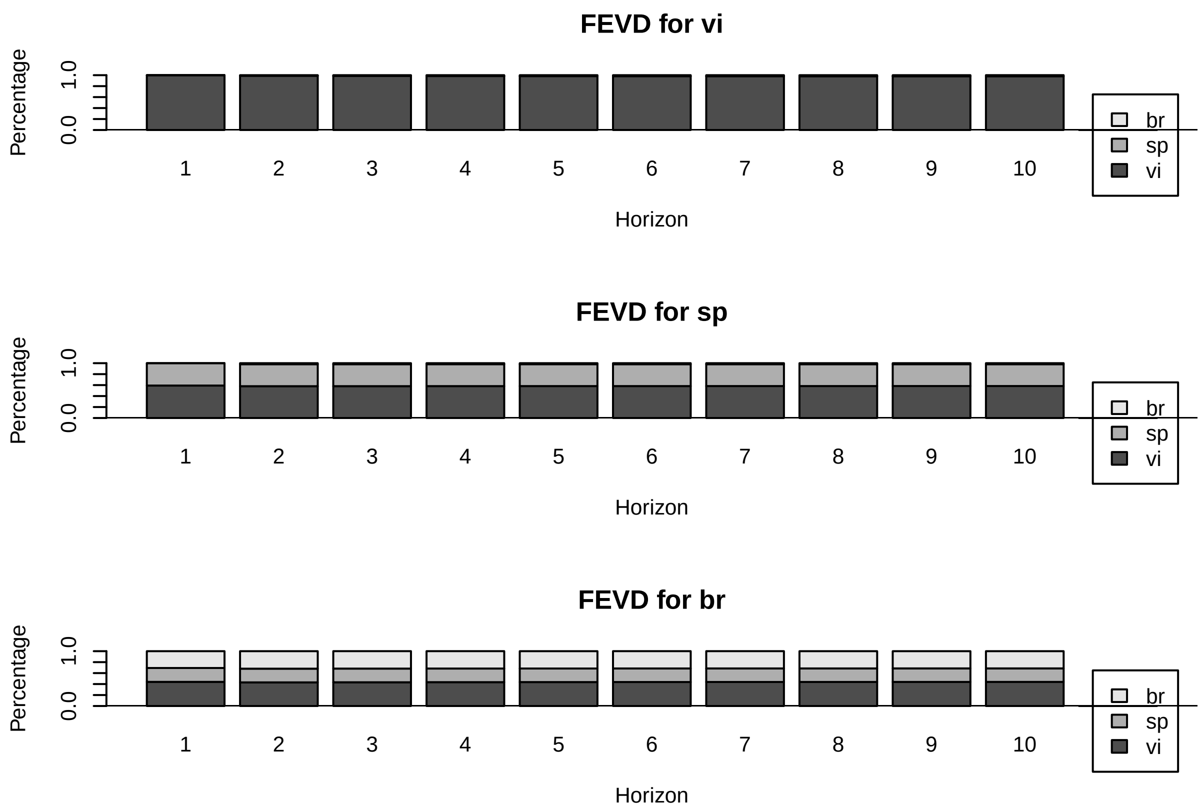
```
causality(var1, cause = c("sp"))$Granger
```

```
##
##  Granger causality H0: sp do not Granger-cause vi br
##
## data:  VAR object var1
## F-Test = 2.7229, df1 = 2, df2 = 1443, p-value = 0.06602
```

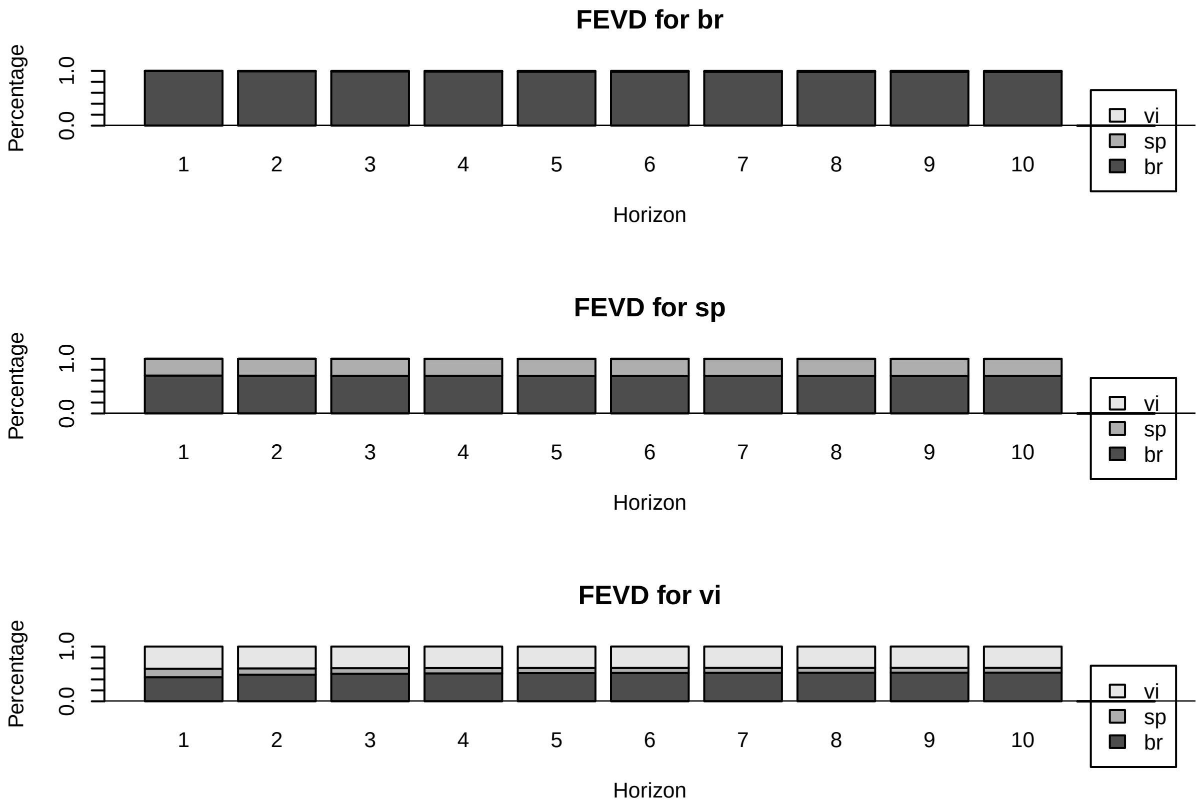
```
causality(var1, cause = c("vi"))$Granger
```

```
##
##  Granger causality H0: vi do not Granger-cause sp br
##
## data:  VAR object var1
## F-Test = 4.5573, df1 = 2, df2 = 1443, p-value = 0.01064
```

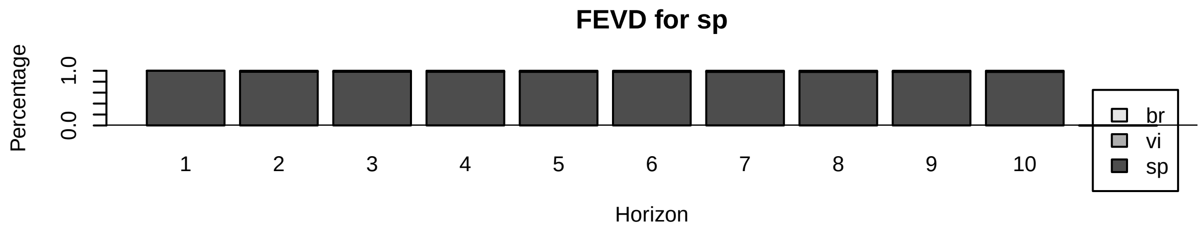
```r
ir <- irf(var1, n.ahead = 10)

vd <- fevd(var1, n.ahead = 10)
plot(vd)
```

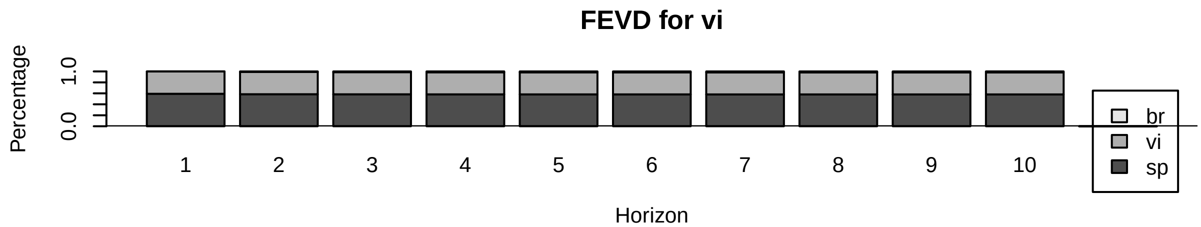


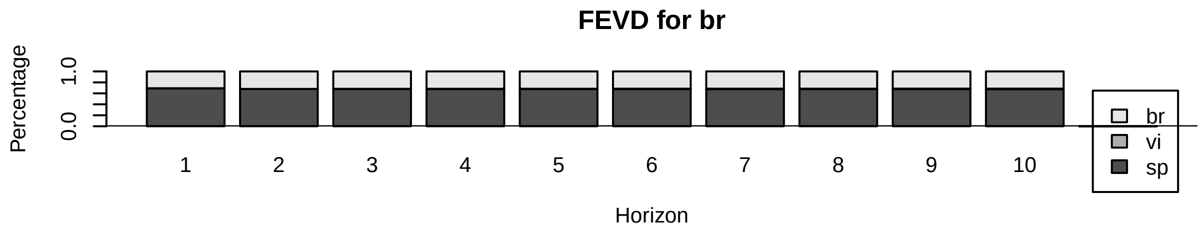

```r
var1_reverse1 <- vars::VAR(train[c("br","sp","vi")], p = 1)
vd_reverse1 <- fevd(var1_reverse1, n.ahead = 10)
plot(vd_reverse1)
```




```
var1_reverse2 <- vars::VAR(train[c("sp","vi","br")], p = 1)
vd_reverse2 <- fevd(var1_reverse2, n.ahead = 10)
plot(vd_reverse2)
```



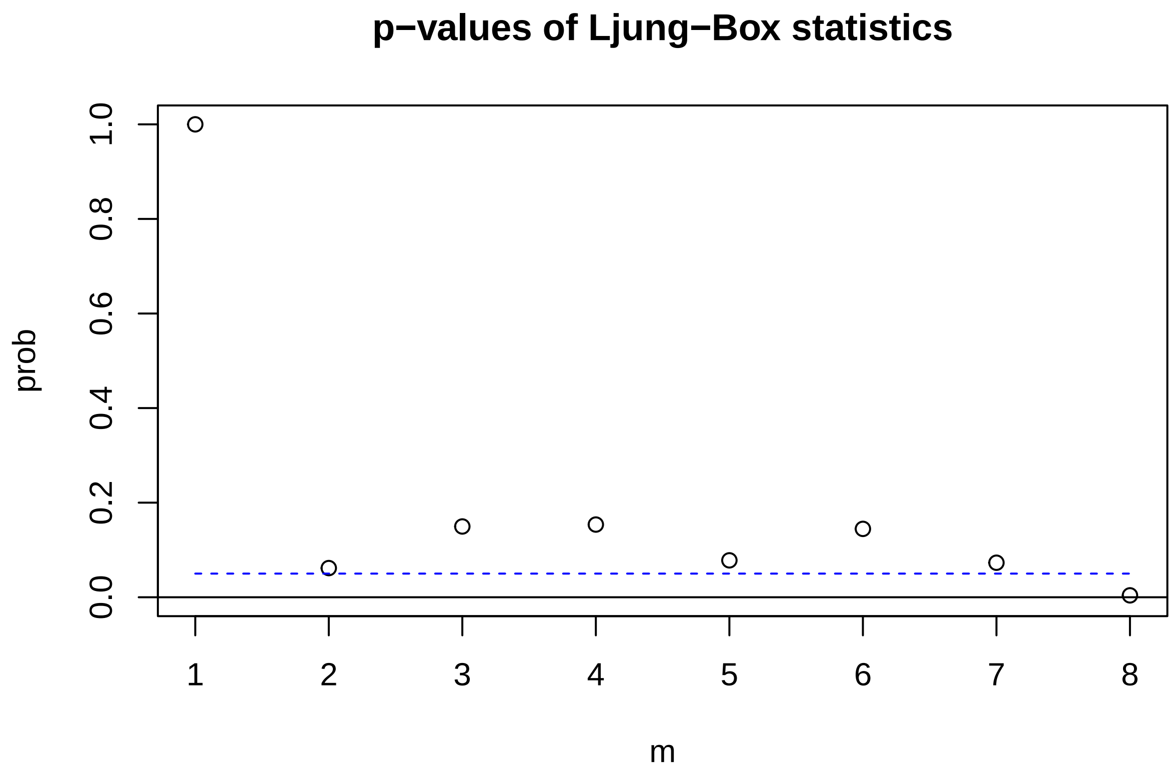




In this section, we assess the adequacy of the models by checking the residual cross-correlation. This allows us to verify if the residuals are uncorrelated across variables, ensuring that the models appropriately capture the relationships in the data. Additionally, we compare the models using AIC, BIC, and RMSE to determine the best-fitting model.

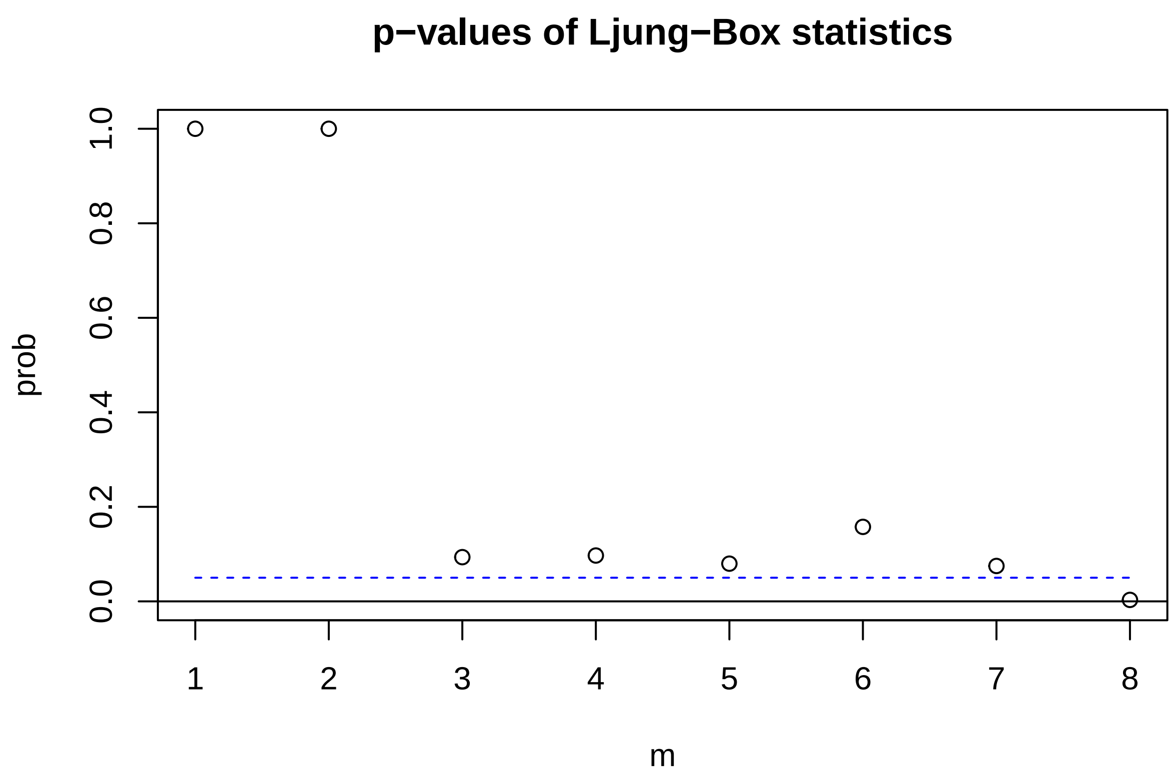
```r
res <- cbind(var1$varresult$vi$residuals,
             var1$varresult$sp$residuals,
             var1$varresult$br$residuals)
mq(res, adj = 9, lag = 8)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
## [1,]   1.00       5.29    0.00     1.00
## [2,]   2.00      16.26    9.00     0.06
## [3,]   3.00      24.17   18.00     0.15
## [4,]   4.00      34.44   27.00     0.15
## [5,]   5.00      48.62   36.00     0.08
## [6,]   6.00      55.07   45.00     0.14
## [7,]   7.00      69.78   54.00     0.07
## [8,]   8.00      96.80   63.00     0.00
```



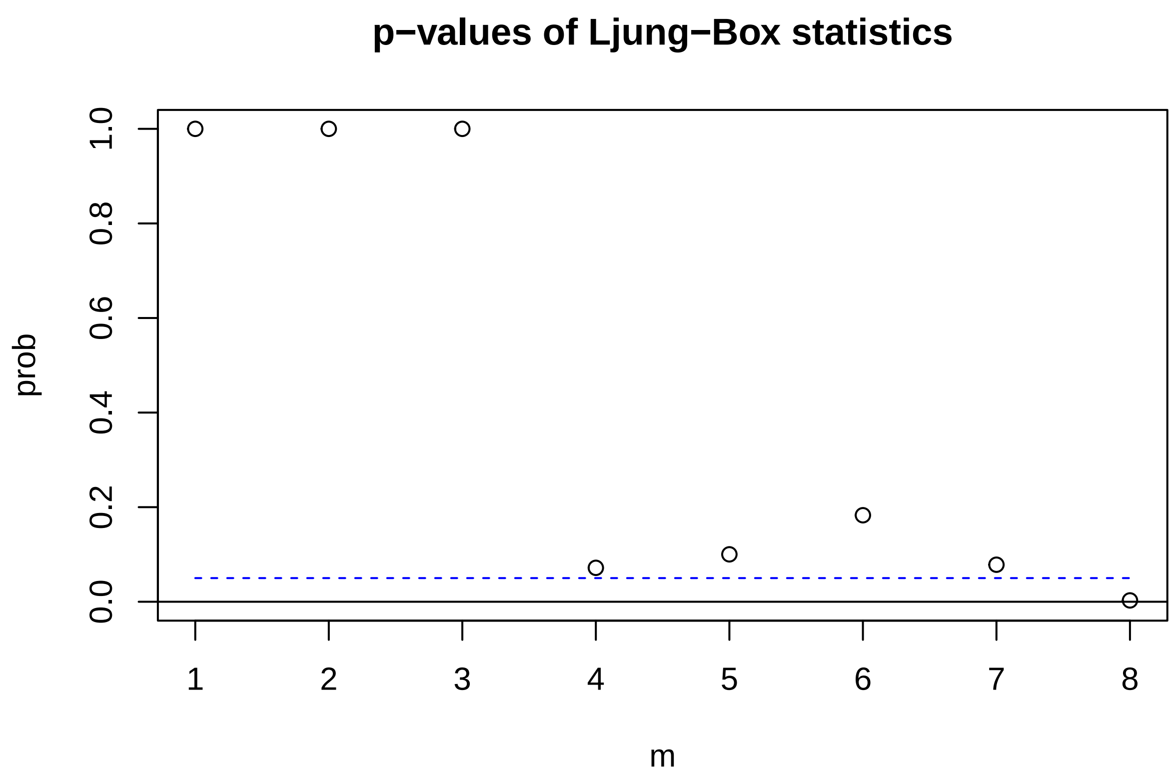

```r
var2 <- vars::VAR(train[c("vi","sp","br")], p = 2)
res2 <- cbind(var2$varresult$vi$residuals,
              var2$varresult$sp$residuals,
              var2$varresult$br$residuals)
mq(res2, lag = 8, adj = 18)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
## [1,]   1.000     0.155  -9.000     1.00
## [2,]   2.000     8.590   0.000     1.00
## [3,]   3.000    14.906   9.000     0.09
## [4,]   4.000    26.123  18.000     0.10
## [5,]   5.000    37.882  27.000     0.08
## [6,]   6.000    44.449  36.000     0.16
## [7,]   7.000    59.298  45.000     0.07
## [8,]   8.000    86.803  54.000     0.00
```




```r
var3 <- vars::VAR(train[c("vi","sp","br")], p = 3)
res3 <- cbind(var3$varresult$br$residuals,
              var3$varresult$sp$residuals,
              var3$varresult$vi$residuals)
mq(res3, lag = 8, adj = 27)
```

```
## Ljung-Box Statistics:
##           m       Q(m)     df    p-value
## [1,]    1.000      0.100 -18.000     1.00
## [2,]    2.000      0.388  -9.000     1.00
## [3,]    3.000      3.140   0.000     1.00
## [4,]    4.000     15.776   9.000     0.07
## [5,]    5.000     25.980  18.000     0.10
## [6,]    6.000     33.441  27.000     0.18
## [7,]    7.000     48.596  36.000     0.08
## [8,]    8.000     75.723  45.000     0.00
```




```r
cbind(AIC(var1), AIC(var2), AIC(var3))
```

```
##          [,1]     [,2]     [,3]
## [1,] 6503.83 6498.901 6493.268
```

```r
cbind(BIC(var1), BIC(var2), BIC(var3))
```

```
##          [,1]     [,2]     [,3]
## [1,] 6554.04 6586.725 6618.668
```

```r
pred1 <- predict(var1, n.ahead = nrow(test))
pred2 <- predict(var2, n.ahead = nrow(test))
pred3 <- predict(var3, n.ahead = nrow(test))

rmse1_br <- sqrt(mean((test$br - pred1$fcst$br[, "fcst"])^2,
                      na.rm = TRUE))
rmse2_br <- sqrt(mean((test$br - pred2$fcst$br[, "fcst"])^2,
                      na.rm = TRUE))
rmse3_br <- sqrt(mean((test$br - pred3$fcst$br[, "fcst"])^2,
                      na.rm = TRUE))

cbind(rmse1_br, rmse2_br, rmse3_br)
```

```
##      rmse1_br rmse2_br rmse3_br
## [1,] 3.236386 3.233722 3.237856
```

Since the RMSE values are similar but the number of parameters differs, we focus on the BIC. The VAR model with p=1p=1 has the lowest BIC, making it the preferred model for its better balance between fit and simplicity.

```r
var1 <- MTS::VAR(train[c("vi","sp","br")], p = 1)
```

```
## Constant term:
## Estimates:  2.155753 -0.3696397 -0.9954587
## Std.Error:  0.4517173 0.3491416 0.4435307
## AR coefficient matrix
## AR( 1 )-matrix
##        [,1]   [,2]   [,3]
## [1,] 0.8820  0.254 -0.246
## [2,] 0.0291 -0.133  0.195
## [3,] 0.0623 -0.153  0.249
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0229 0.1093 0.0833
## [2,] 0.0177 0.0845 0.0644
## [3,] 0.0224 0.1073 0.0818
##
## Residuals cov-mtx:
##           [,1]      [,2]      [,3]
## [1,] 12.087521 -7.188117 -7.872525
## [2,] -7.188117  7.221161  7.630284
## [3,] -7.872525  7.630284 11.653362
##
## det(SSE) =  127.3377
## AIC =  4.88388
## BIC =  4.961402
## HQ  =  4.914336
```

```r
refvar1 <- refVAR(var1)
```

```
## Constant term:
## Estimates:  2.155753 -0.3696397 -0.9954587
## Std.Error:  0.4517173 0.3491416 0.4435307
## AR coefficient matrix
## AR( 1 )-matrix
##        [,1]   [,2]   [,3]
## [1,] 0.8820  0.254 -0.246
## [2,] 0.0291 -0.133  0.195
## [3,] 0.0623 -0.153  0.249
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0229 0.1093 0.0833
## [2,] 0.0177 0.0845 0.0644
## [3,] 0.0224 0.1073 0.0818
##
## Residuals cov-mtx:
##           [,1]      [,2]      [,3]
## [1,] 12.087521 -7.188117 -7.872525
## [2,] -7.188117  7.221161  7.630284
```

```
## [3,] -7.872525  7.630284 11.653362
##
## det(SSE) =  127.3377
## AIC =  4.88388
## BIC =  4.961402
## HQ  =  4.914336
```

We checked whether any parameters were insignificant and could be removed, but none of them were deemed non-significant. Therefore, all parameters remain in the model.

Forecast

Now, we proceed with forecasting for the next week using the model we have estimated. We use all available data for forecasting, unlike before when we split the data into training and testing sets.

```r
rawdata = read_excel(here("blackrock.xlsx"), sheet = 8, col_names = TRUE)
colnames(rawdata) <- c( "Date", "br", "sp","vi")

last <- tail(rawdata$br, 1)
log_last <- log(last)

var1 <- MTS::VAR(data[c("vi", "sp", "br")], p = 1)
```

```
## Constant term:
## Estimates:  2.341816 -0.3756059 -0.9277752
## Std.Error:  0.4484359 0.3357557 0.4295387
## AR coefficient matrix
## AR( 1 )-matrix
##        [,1]   [,2]   [,3]
## [1,] 0.8720  0.231 -0.192
## [2,] 0.0307 -0.126  0.168
## [3,] 0.0600 -0.140  0.207
## standard error
##        [,1]  [,2]   [,3]
## [1,] 0.0228 0.107 0.0811
## [2,] 0.0171 0.080 0.0607
## [3,] 0.0218 0.102 0.0777
##
## Residuals cov-mtx:
##           [,1]      [,2]      [,3]
## [1,] 12.654176 -7.159248 -7.833785
## [2,] -7.159248  7.093812  7.549141
## [3,] -7.833785  7.549141 11.610146
##
## det(SSE) =  137.4066
## AIC =  4.956278
## BIC =  5.027804
## HQ  =  4.984251
```

```r
for1 <- VARpred(var1, h= 1)
```

```
## orig  540
```

```
## Forecasts at origin:  540
##      vi      sp      br
## 22.9751  0.3359  0.4849
## Standard Errors of predictions:
## [1] 3.557 2.663 3.407
## Root mean square errors of predictions:
## [1] 3.570 2.673 3.420
```

```r
br_logret1 <- for1$pred[3]

log_forecast <- log_last + (br_logret1 / 100)

forecast_price <- exp(log_forecast)
print(forecast_price)
```

```
##       br
## 924.8342
```

```r
real_price <- 967.06

abs_error1 <- abs(real_price - forecast_price)
perc_error1 <- (abs_error1 / real_price) * 100

cat("Absolute error:", abs_error1, "\n")
```

```
## Absolute error: 42.22579
```

```r
cat("Percentage error:", perc_error1, "%\n\n")
```

```
## Percentage error: 4.366408 %
```