Microsoft Stock Price Analysis: Univariate Modeling and Forecasting (2017-2022)

Chan Wai Wong

December 6, 2023

Contents

Introduction

This study focuses on conducting a fundamental time-series analysis by univariate modeling of the closing stock prices of Microsoft spanning from December 31, 2017, to December 31, 2022. The objective is to gain insights into the time-series components and discern the underlying patterns, including seasonality, trends, and stationarity within the data. Subsequently, the study aims to initiate the construction of classical time-series models, specifically Exponential Smoothing (ETS) and Autoregressive Integrated Moving Average (ARIMA), to forecast the raw stock prices for the upcoming year, 2023. The research endeavors to introduce the mathematical expressions underpinning these models to facilitate a comprehensive understanding of their conceptual frameworks. Additionally, the study delves into the development of several Generalized AutoRegressive Conditional Heteroscedasticity (GARCH) models to analyze and comprehend volatility, presenting forecast results for volatility in the process.

Data

```
getSymbols("MSFT",
           from = "2017-12-31",
           to = "2022-12-31")
```

```
## [1] "MSFT"
```

```
class(MSFT)
```

```
## [1] "xts" "zoo"
```

```
head(MSFT)
```

```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2018-01-02     86.13     86.31    85.50      85.95    22483800      80.22898
## 2018-01-03     86.06     86.51    85.97      86.35    26061400      80.60238
## 2018-01-04     86.59     87.66    86.57      87.11    21912000      81.31177
## 2018-01-05     87.66     88.41    87.43      88.19    23407100      82.31992
## 2018-01-08     88.20     88.58    87.60      88.28    22113000      82.40393
## 2018-01-09     88.65     88.73    87.86      88.22    19484300      82.34791
```

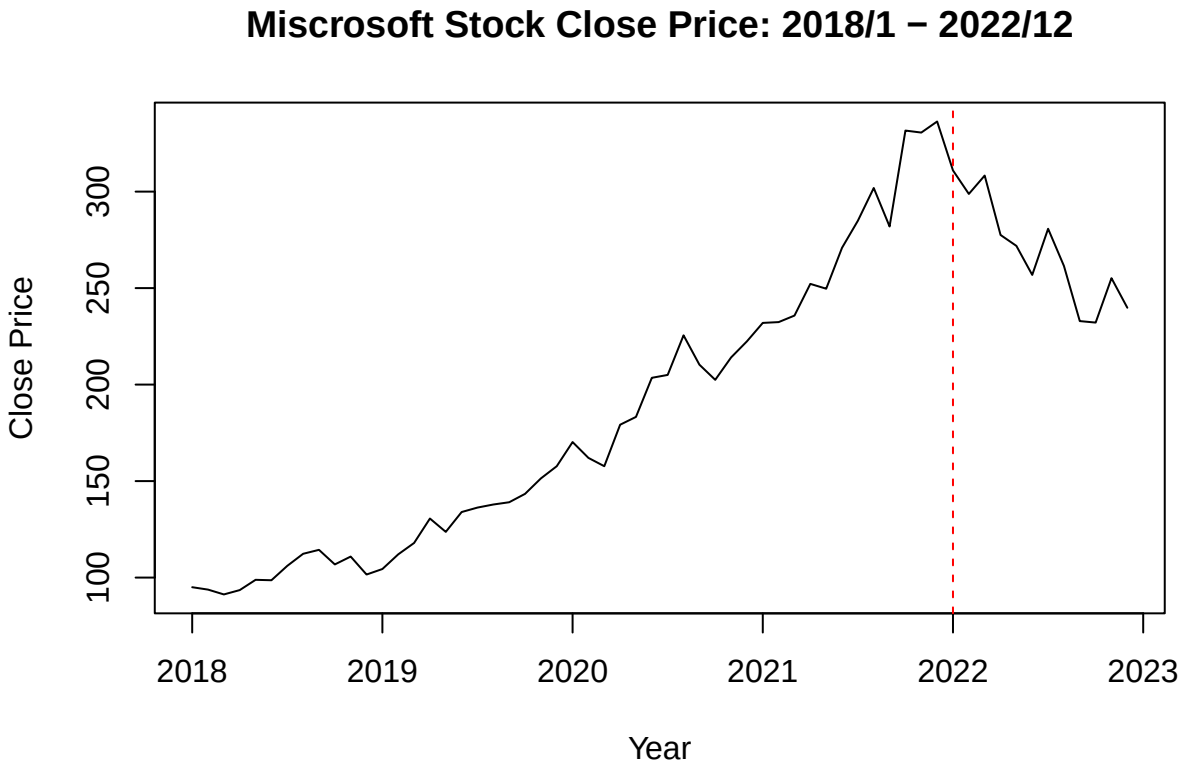
Variables Descriptions:

- Time-Stamp
- Open: Price at market opening time.
- High: Highest price during the trading day.
- Low: Lowest price during the trading day.
- Close: Price at market closing time.
- Volume: Number of shares of the stock traded during the trading day.
- Adjusted: The closing price after adjustments for all corporate actions.

Exploratory Data Analysis (EDA)

```
MSFT_monthly <- to.monthly(MSFT)

# Converting xts to ts object
ts <- ts(MSFT_monthly$MSFT.Close,
         start = c(2018, 1),
         end = c(2022, 12),
         frequency = 12)

# Time plot
```

```r
plot(ts,
     main = "Miscrosoft Stock Close Price: 2018/1 - 2022/12",
     xlab = "Year", ylab = "Close Price")
abline(v = "2022", col = "red", lty = 2)
```



The Time plot illustrates a linear uptrend from 2018 to late 2021, followed by a subsequent downtrend leading into 2023.

```r
# Train Test Split
training <- window(ts, start = c(2018, 1), end = c(2021, 12))
test <- window(ts, start = c(2022, 1), end = c(2022, 12))
```

Classical Time-Series Decomposition Analysis

- Application: It helps to decompose a time series into several components, which is useful to further understand how a time series behavior by recognizing their potential underlying pattern(s).

Time series data refers to a collection of random variables that are organized and indexed based on the order in which they are observed over time.

Addictive Decomposition:

$$y_t = S_t + T_t + R_t$$

Multiplicative Decomposition:

$$y_t = S_t \times T_t \times R_t$$

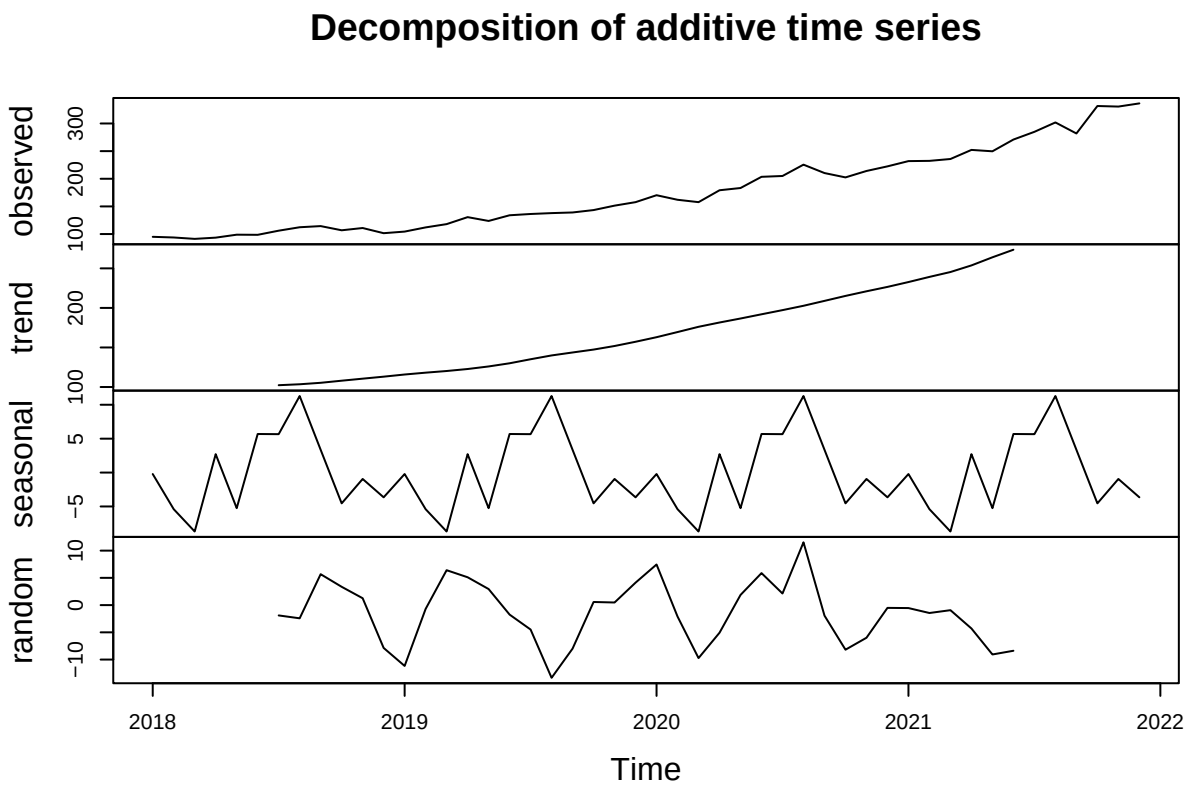
Where:

- y_t denotes Data

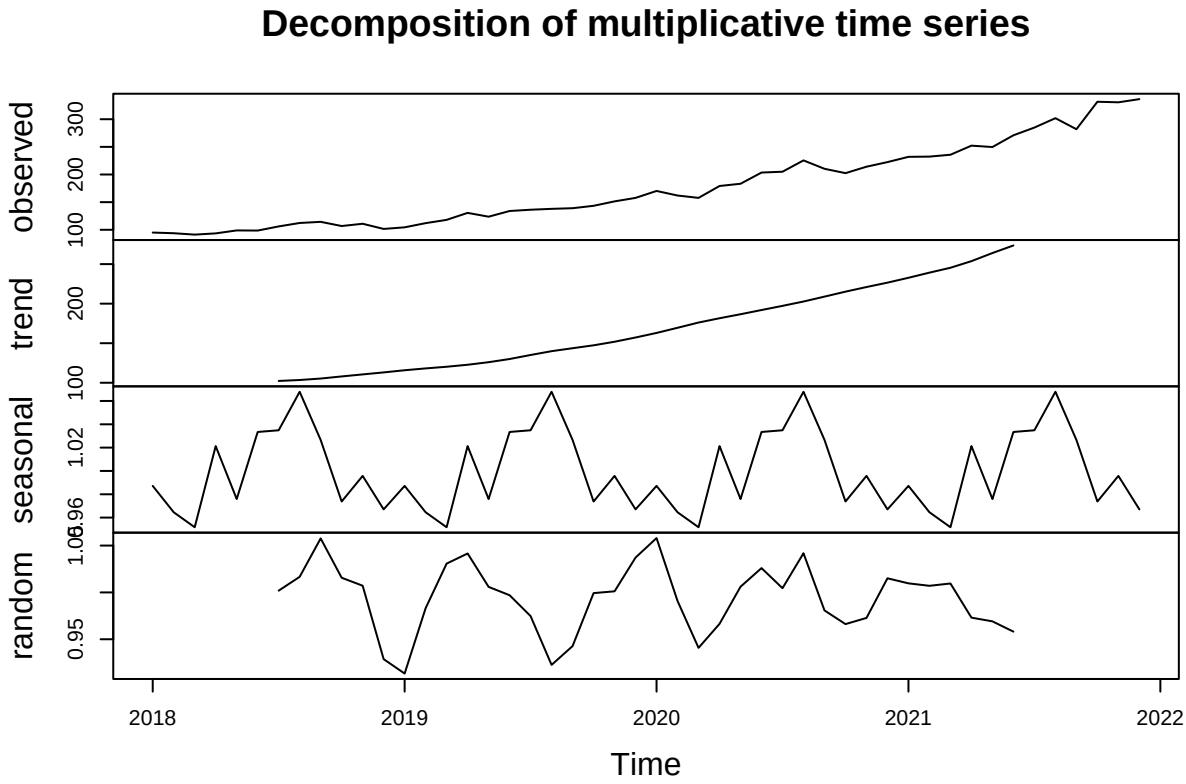
- S_t denotes Seasonal Variation
- T_t denotes Trend plus Cycle
- R_t denotes Residual or Error

The assumption of classical decomposition methods is that the seasonal component repeats itself in a similar pattern from year to year, indicating there is a regular, repeating pattern of seasonality in the time series data.

```
add_dcomp <- decompose(training, type = "add")
mult_dcomp <- decompose(training, type = "mult")

# Plots
plot(add_dcomp)
```




```
plot(mult_dcomp)
```

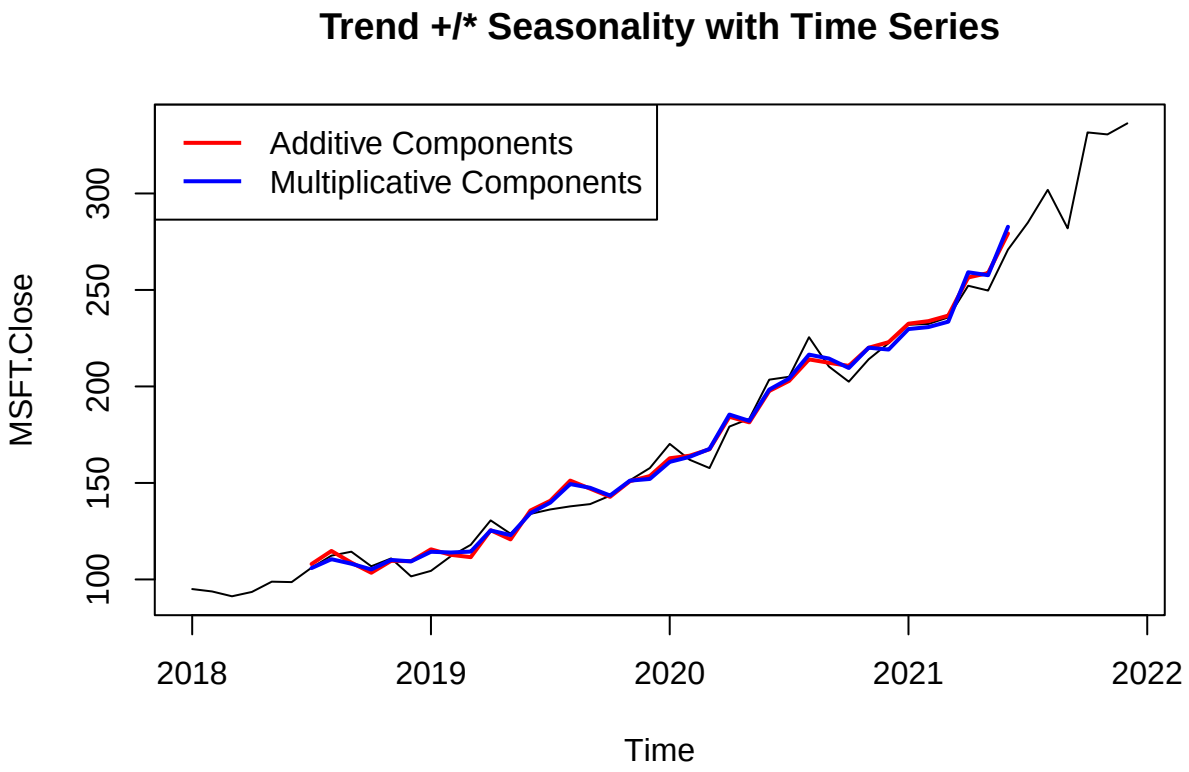



```
# Extracting components
add_trend <- add_dcomp$trend
add_seasonal <- add_dcomp$seasonal
add_random <- add_dcomp$random

mult_trend <- mult_dcomp$trend
mult_seasonal <- mult_dcomp$seasonal
mult_random <- mult_dcomp$random
```

To observe whether the seasonal variation is relatively constant or increase over time.

Approach 1: Comparing the Additive and Multiplicative assumptions in the decomposition of a time series.

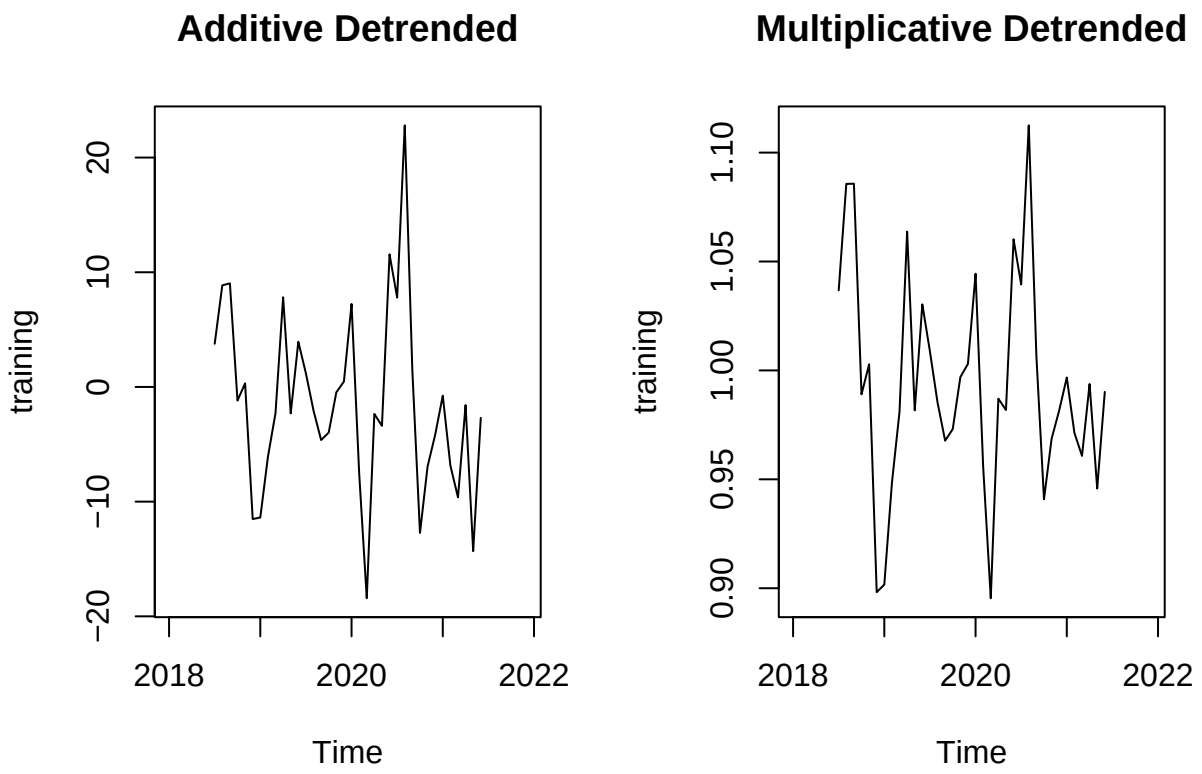
```
plot.ts(training,
        main = "Trend +/* Seasonality with Time Series")
lines(add_trend + add_seasonal, col = "red", lwd = 2)
lines(mult_trend * mult_seasonal, col = "blue", lwd = 2)
legend("topleft",
       legend = c("Additive Components", "Multiplicative Components"),
       col = c("red", "blue"),
       lwd = 2)
```



It is not easy to distinguish whether the addictive or the multiplicative decomposition models without error terms follows the original time series data closely. Noticed that a drawback in the decomposition methods, as they generate missing values for the initial and final data points of the series. This occurs because these methods employ a moving average to estimate the trend and cyclical components.

Approach 2: The detrended data emphasizes the seasonal variations of the time series.

```
detrend_add <- training - add_trend
detrend_mult <- training / mult_trend

par(mfrow = c(1, 2))
plot.ts(detrend_add, main = "Additive Detrended")
plot.ts(detrend_mult, main = "Multiplicative Detrended")
```

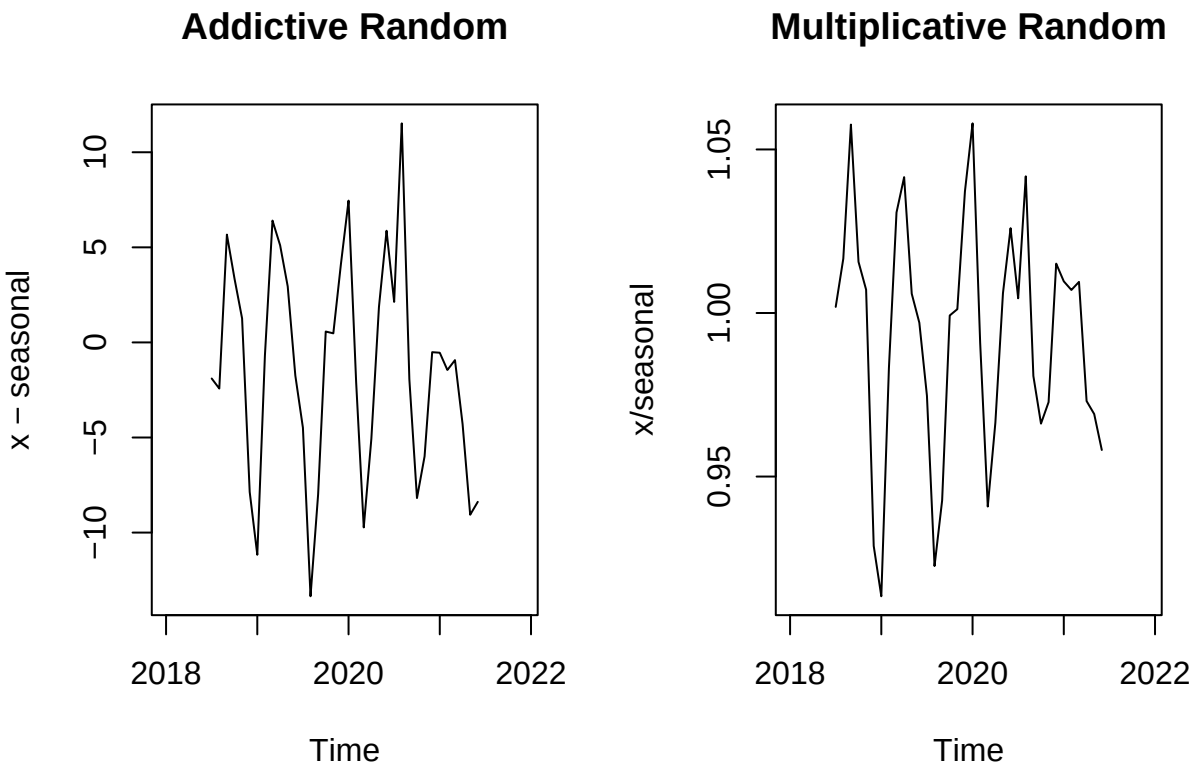


- The detrended data should centered at 0 as adding 0 to the addictive decomposition model won't change the model.
- The detrended data should centered at 1 as multiplying 1 to the multiplicative decomposition model won't change the model.

Both detrend data show similar fluctuation levels and no regular seasonality or patterns are observed. It suggest that the underlying patterns in the data are relatively random and do not follow a consistent trend or seasonal behavior.

Approach 3: Observing the random noise component, the lower the better.

```
par(mfrow = c(1, 2))
plot(add_random, main = "Addictive Random")
plot(mult_random, main = "Multiplicative Random")
```



It is difficult to determine which one has better consistent in the variability.

Conclusion

Neither an additive nor a multiplicative decomposition model would be suitable for capturing meaningful patterns in the data. This suggests that the time series is essentially stochastic, and classic decomposition methods may not be effective. Therefore, exploring advanced statistical approaches for analyzing the data is essential, such as utilizing machine learning algorithms like Exponential Smoothing or ARIMA models. To be more specific, there are other machine learning algorithms designed to handle noisy or irregular patterns effectively.

Limitations & Recommendations

- 1. The `ts()` function does have limitations when dealing with stock daily data due to the `frequency` parameter.
  - The `xts` object is more flexible than the `ts` object without being restricted by the `frequency` parameter.

Exponential Smoothing (ETS)

- Application: It is simple, computation efficient, and useful in short-term forecasting. Simple Exponential Smoothing is good for data with no clear trend or seasonal pattern while its extension part, Holt's linear/Holt-Winters' seasonal methods are good for data exhibit trend or/and seasonal patterns.

Simple Exponential Smoothing (SES)

The idea behind SES forecasting involves using a weighted averages approach where greater emphasis is placed on recent observations. This emphasis is achieved by assigning higher weights to more recent data points, while the weights decrease exponentially as the observations become older.

The one-step-ahead forecast equation:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \dots$$

Where:

- α denotes a smoothing parameter between 0 and 1

Two equivalent forms of the forecast equation:

Weighted average form:

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_0$$

Where:

- ℓ_0 denotes the first fitted value at time 1

Component form:

This form comprises a forecast equation and a smoothing equation for each of the components included in the method.

Forecast equation:

$$\hat{y}_{t+h|t} = \ell_t$$

Smoothing equation:

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

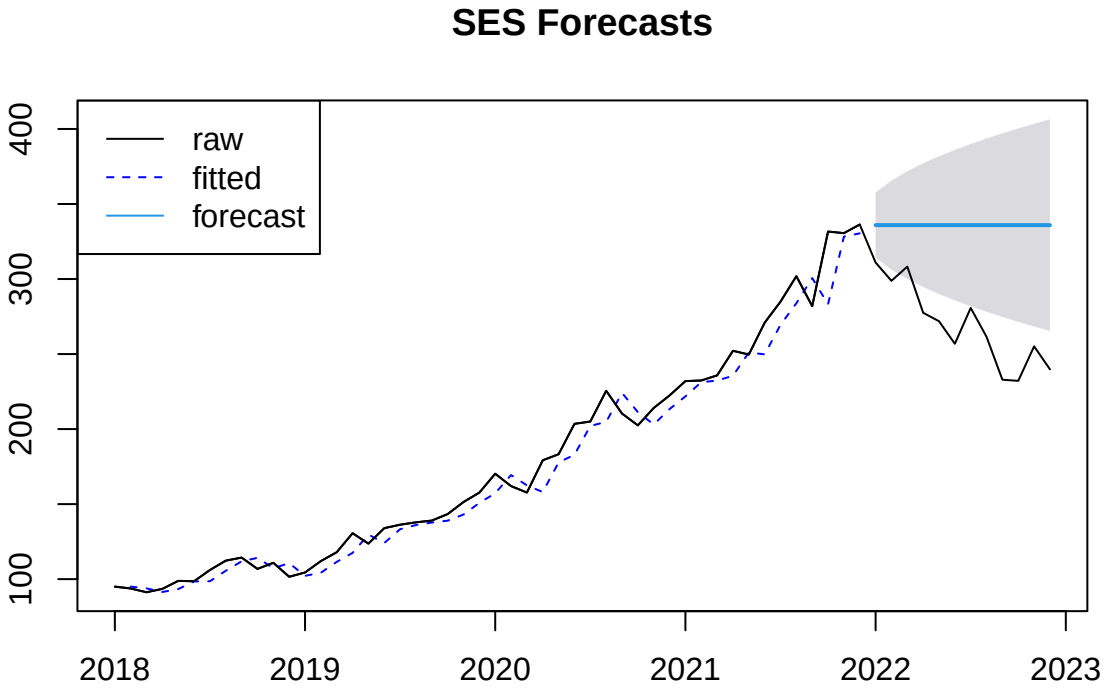
Where:

- α denotes a smoothing parameter for the level
- ℓ_t denotes the smoothed value of the series at time t
- h can be any arbitrary value for getting the fitted values

The forecast equation indicates that the predicted value at time $t+1$ corresponds to the estimated level at time t.

The smoothing equation indicates that the estimated level of the series at each period t.

```
smoothing_1 <- HoltWinters(training, beta = F, gamma = F)
hw_fcast_1 <- forecast(smoothing_1, h = 12, level = 95)
plot(hw_fcast_1, main = "SES Forecasts")
lines(hw_fcast_1$fitted, lty = 2, col = "blue")
lines(ts)
legend("topleft",
       legend = c("raw", "fitted", "forecast"),
       lty = c(1, 2, 1),
       col = c("black", "blue", "#2297E6"))
```



Holt's linear trend method

Holt's linear trend method is the extended simple exponential smoothing including trend component.

This form comprises a forecast equation and two smoothing equations (level + trend) for each of the components included in the method.

Forecast equation:

$$\hat{y}_{t+h|t} = \ell_t + hb_t$$

Level equation:

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

Trend equation:

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$

Where:

- α denotes a smoothing parameter for the level
- β^* denotes a smoothing parameter for the trend
- ℓ_t denotes the smoothed value of the series at time t
- h can be any arbitrary value for getting the fitted values

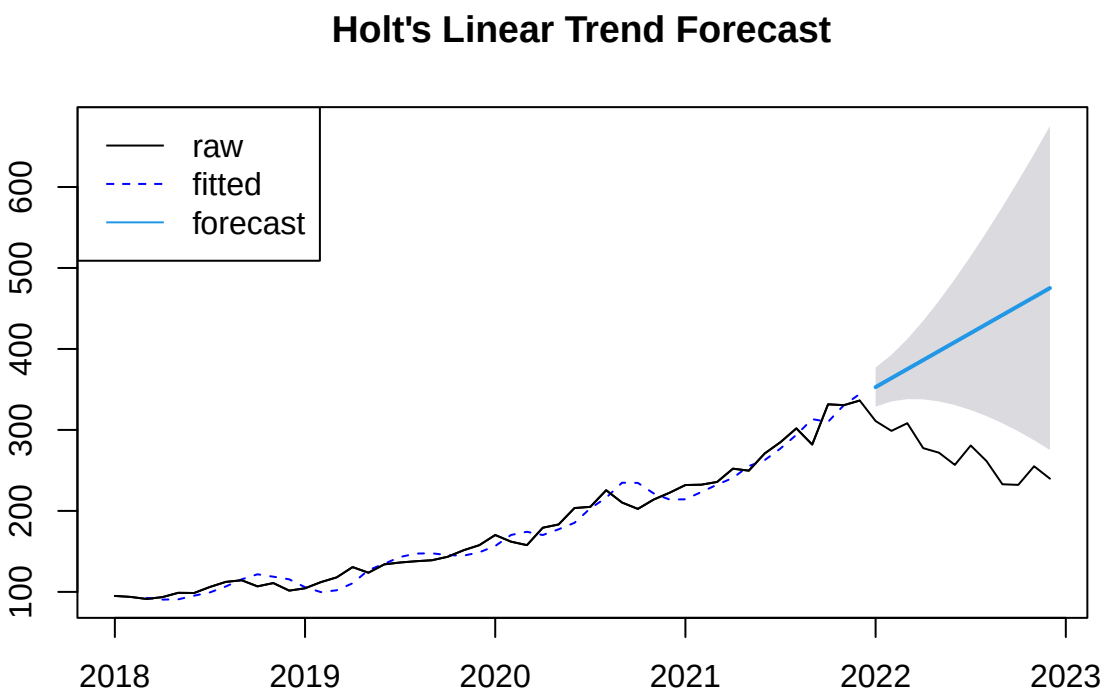
The forecast equation indicates that the predicted value at time $t+1$ corresponds to the estimated level at time t.

The level equation indicates that the ℓ_t is calculated as a weighted average of observation y_t and the one-step-ahead training forecast for time t, represents as $\ell_{t-1} + b_{t-1}$.

The trend equation indicates that the b_t is calculated as a weighted average of the estimated trend at time t based on previous estimate of the trend, represents as $\ell_t - \ell_{t-1}$ and b_{t-1}.

```
smoothing_2 <- HoltWinters(training, beta = T, gamma = F)
hw_fcast_2 <- forecast(smoothing_2, h = 12, level = 95)
plot(hw_fcast_2, main = "Holt's Linear Trend Forecast")
```

```r
lines(hw_fcast_2$fitted, lty = 2, col = "blue")
lines(ts)
legend("topleft",
       legend = c("raw", "fitted", "forecast"),
       lty = c(1, 2, 1),
       col = c("black", "blue", "#2297E6"))
```



Holt-Winters' seasonal methods

Holt-Winters' seasonal methods (additive + multiplicative) is the extended simple exponential smoothing including seasonality component.

This form comprises a forecast equation and three smoothing equations (level + trend + seasonal) for each of the components included in the method.

Additive

Forecast equation:

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)}$$

Level equation:

$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

Trend equation:

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$

Seasonal equation:

$$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m})$$

Where:

- α denotes a smoothing parameter for the level
- β^* denotes a smoothing parameter for the trend

- γ denotes a smoothing parameter for the seasonal
- m denotes the frequency of the seasonality
- k denotes the integer part of $\frac{(h-1)}{m}$, ensuring that the seasonal indices used for forecasting are derived from the last year of the sample data.
- ℓ_t denotes the smoothed value of the series at time t
- h can be any arbitrary value for getting the fitted values

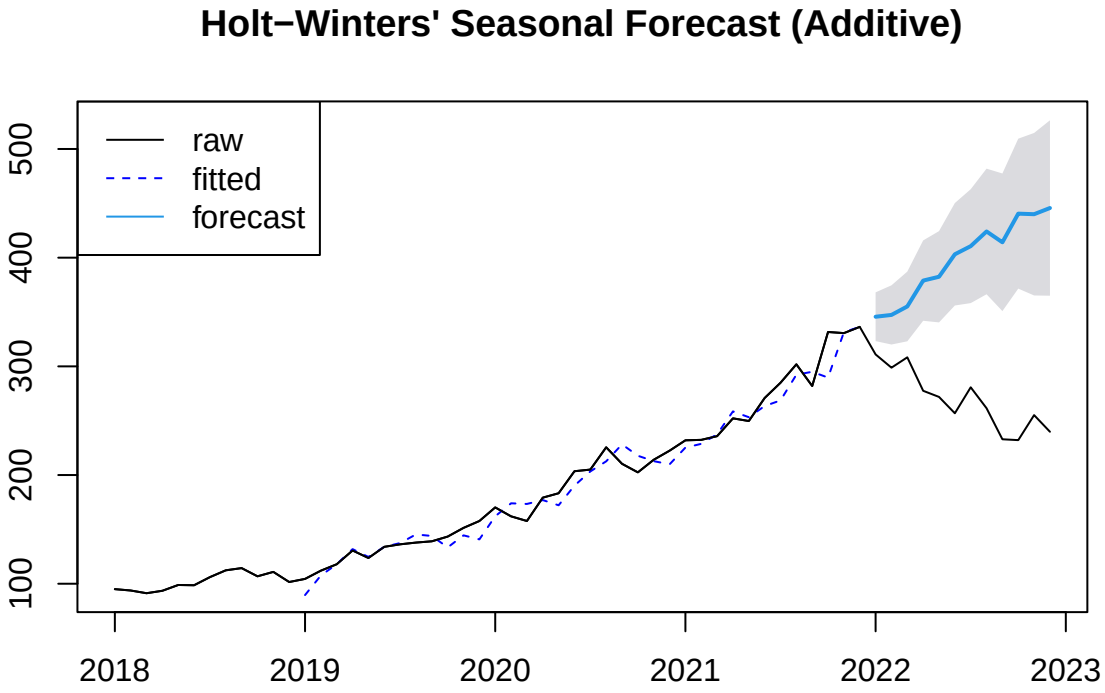
The forecast equation indicates that the predicted value at time $t+1$ corresponds to the estimated level at time t.

The level equation indicates that a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t.

The trend equation indicates that the b_t is calculated as a weighted average of the estimated trend at time t based on previous estimate of the trend, represents as $\ell_t - \ell_{t-1}$ and b_{t-1} *(SAME AS Holt's linear trend method)*.

The seasonal equation indicates that the seasonal component s_t at time t is calculated based on the difference between the observed value y_t and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$, adjusted for the seasonal indices s_{t-m} from the same season in the previous year.

```
smoothing_3 <- HoltWinters(training, seasonal = "add")
hw_fcast_3 <- forecast(smoothing_3, h = 12, level = 95)
plot(hw_fcast_3, main = "Holt-Winters' Seasonal Forecast (Additive)")
lines(hw_fcast_3$fitted, lty = 2, col = "blue")
lines(ts)
legend("topleft",
       legend = c("raw", "fitted", "forecast"),
       lty = c(1, 2, 1),
       col = c("black", "blue", "#2297E6"))
```



Multiplicative

Forecast equation:

$$\hat{y}_{t+h|t} = (\ell_t + hb_t)s_{t+h-m(k+1)}$$

Level equation:

$$\ell_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

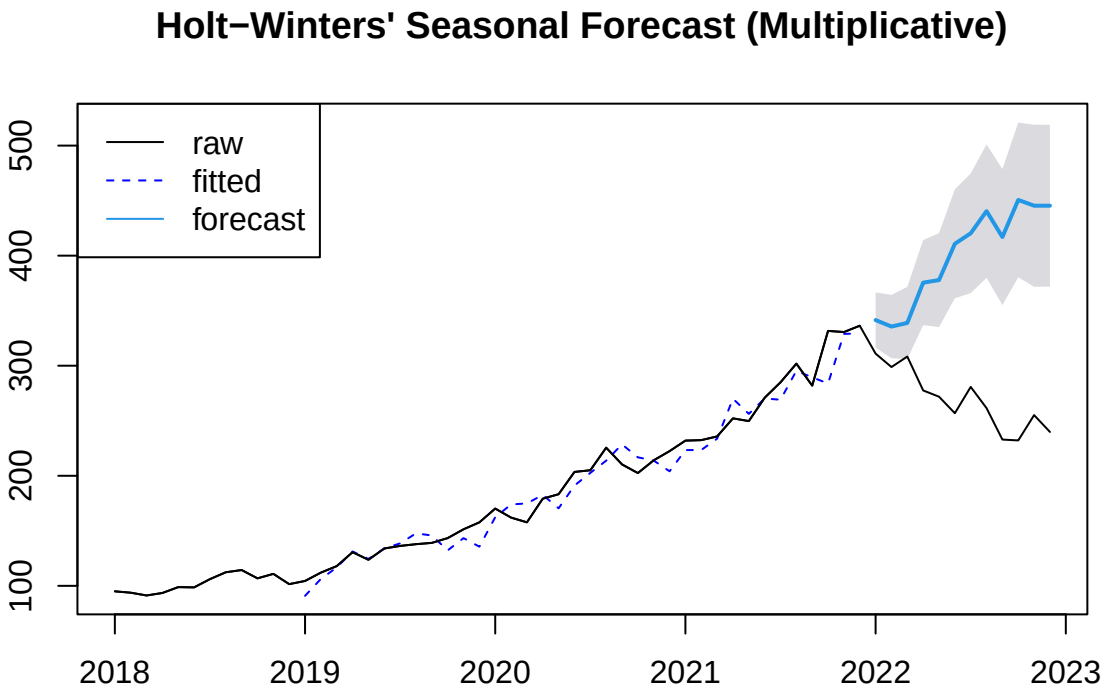
Trend equation:

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$

Seasonal equation:

$$s_t = \gamma \frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m})$$

```
smoothing_4 <- HoltWinters(training, seasonal = "mult")
hw_fcast_4 <- forecast(smoothing_4, h = 12, level = 95)
plot(hw_fcast_4, main = "Holt-Winters' Seasonal Forecast (Multiplicative)")
lines(hw_fcast_4$fitted, lty = 2, col = "blue")
lines(ts)
legend("topleft",
       legend = c("raw", "fitted", "forecast"),
       lty = c(1, 2, 1),
       col = c("black", "blue", "#2297E6"))
```



Model Evaluation based on the Root Mean Squared Error (RMSE)

Method	RMSE
SES	12.12914
Holt's Linear	11.98675
Holt-Winters' (Add)	10.05957
Holt-Winters' (Mult)	11.25920

Conclusion

The Exponential Smoothing methods exhibit poor performance in forecasting this time series data, as evident from both visualization plots and high RMSE values. The predictions for one year ahead diverge significantly

from the actual data, indicating a lack of accuracy. In addition, Holt's linear trend and Holt-Winters' seasonal methods seem heavily reliant on past observations, leading to an increase in trend predictions. Furthermore, all RMSE values are greater than 10, indicating that the average magnitude of the errors between predicted and actual values is large. This finding is consistent with the visualization results.

Limitations & Recommendations

- 1. Although Exponential Smoothing Method does not strictly required a stationary time series, it assumes that the time series has a certain level of smoothness or regularity in its patterns. It makes it challenging to capture complex pattern such as sudden changes or irregular fluctuations.
  - Considering alternative forecasting methods such as ARIMA or other machine learning models which can handle a wider range of patterns and trends.
- 2. It is sensitive to initial conditions which means the choice of the stating values or parameters is sensitive as small variations in the initial conditions can result in substantial differences in the final forecasted values.
  - Performing cross-validation to assess the performance of the model with different initial conditions to evaluate how well the model generalizes to different starting points.

Autoregressive Integrated Moving Average (ARIMA)

- Application: ARIMA is widely used and powerful tool across diverse domains for comprehending and forecasting time-dependent patterns. The primary objective is to describe autocorrelation within the data. In essence, ARIMA models provide interpretable outcomes, aiding in the understanding of the connection between historical and forthcoming price trends. It tends to perform better for short-term forecasting.

In this study, it will be used to forecast the raw values, and help to understand other basic concepts.

ARIMA(p,d,q) model combines three important elements:

- AR: This model captures the relationship where a variable undergoes changes that depend on its own previous values, known as lagged values.
  - **p: the lag order** - the number of lag observations in the model.
- I: This aspect involves differencing the raw observations to achieve stationarity. Stationarity implies that statistical properties, such as mean and variance, remain constant over time. Differencing replaces data values with the differences between these values and their preceding ones.
  - **d: the degree of differencing** - the number of times the raw observations are differenced.
- MA: This component accounts for the correlation between an observation and the residual error derived from a moving average model applied to earlier observations.
  - **q: the order of the moving average** - the size of the moving average window.

Note that: Seasonal ARIMA (SARIMA) model contains the seasonal parts of the model - ARIMA$(p, d, q)(P, D, Q)_m$, where **m** represents the number of observations per year. For example, when m = 4, it is for quarterly data; when m = 12, it is for monthly data (same as the `frequecey` parameter in `ts()` function & `seasonal = list(order = c(P, D, Q), period = m)`).

The procedure of building ARIMA model will be provided in below:

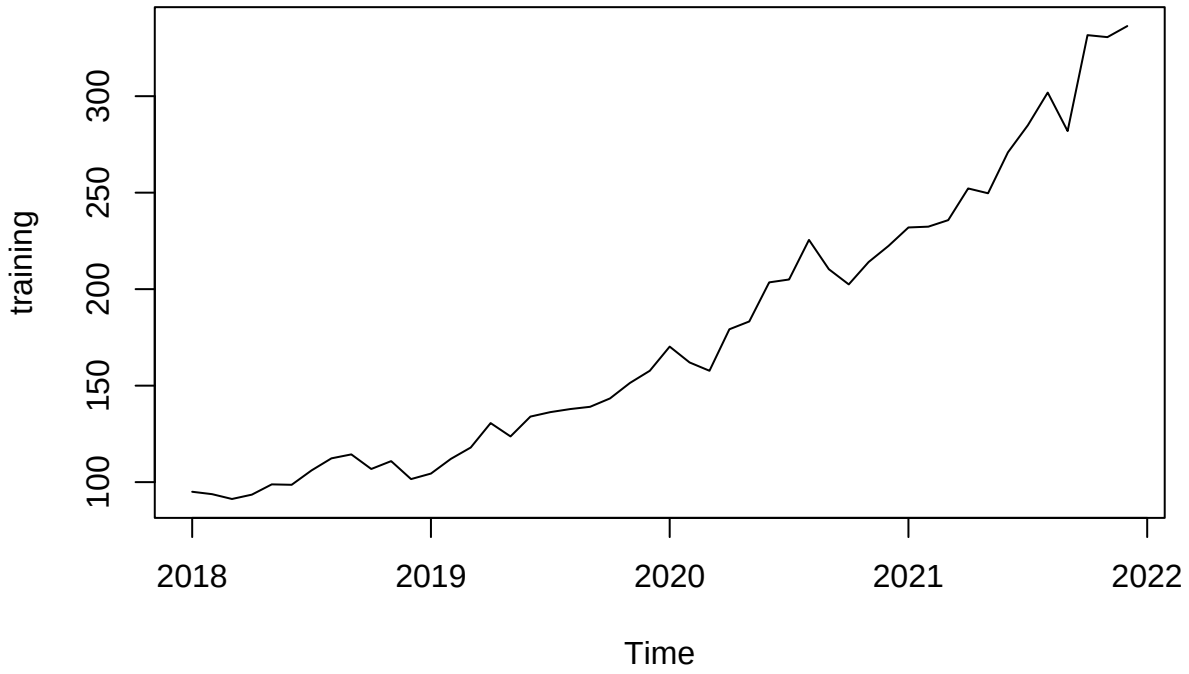
Step 1: Checking Stationarity

A stationary time series means a time series without predictable trend patterns, and having a constant mean and variance over time.

ARIMA models assume stationarity, so it is one of the most important conditions to check before building an ARIMA model.

Approach 1: Observing the time-series plot to see if there are clear trend or seasonal patterns exist.

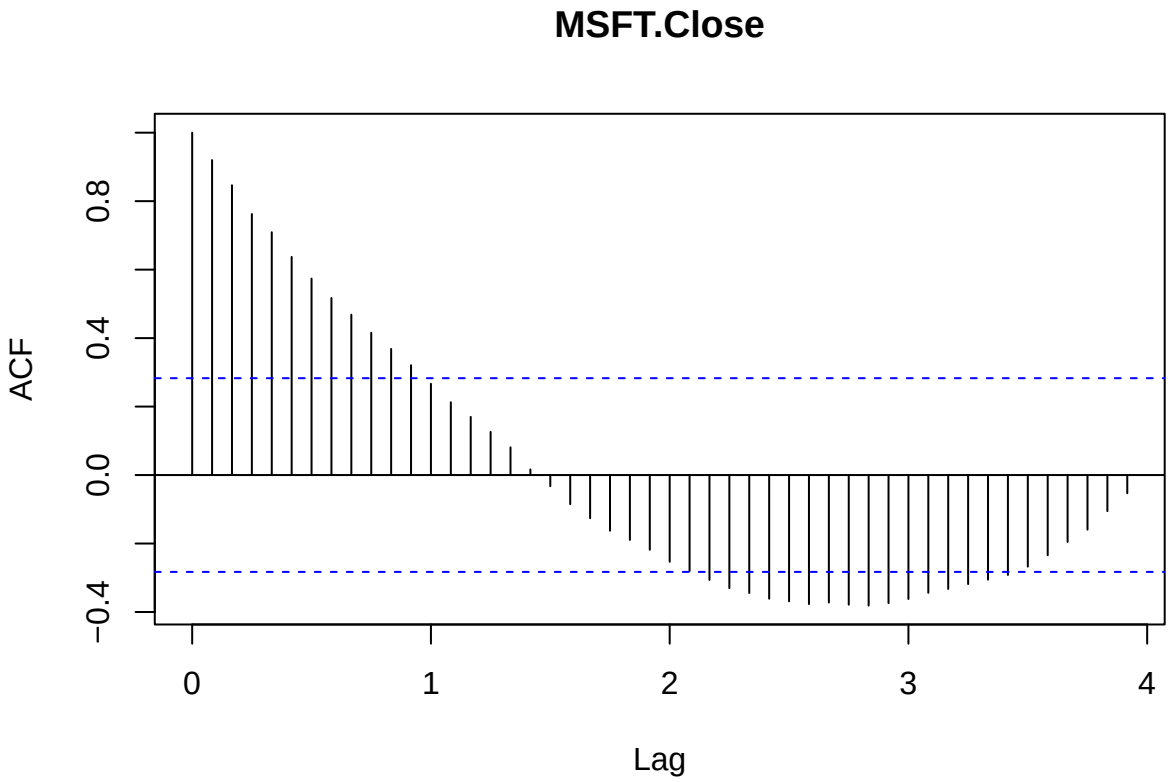
```
ts.plot(training)
```



There is a **clear uptrend** in the time-series which is violating the characteristics of a stationary time-series data. It is **non-stationary**.

Approach 2: Identify if correlation at different time lags goes to 0.

```
acf(training, lag.max = 100)
```



Common ACF plots:

- Takes a long time decay and also looks similar to a sin graph (Trend & Seasonality).
- Decays very slowly and the spikes are out of the blue confidence intervals (Trend).
- Decays quickly and looks similar to a sin graph (Seasonality).

Observing **lags decay slowly, similar to a sin graph, and most of them are exceeding the confidence interval of the ACF**. It provides evidence of a time-dependent structure in the data which indicates the observed autocorrelations are unlikely to be due to random chance. It is **non-stationary**.

Approach 3: Ljung-Box test for independence

- The Ljung-Box test assesses if there are significant correlations at specified lags (commonly 1-25) in a time series. Its null hypothesis assumes independence among observations, with a low p-value signaling evidence against independence. In essence, it helps determine whether observed correlations are meaningful or random.

```
Box.test(training, lag = 25, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  training
## X-squared = 273.65, df = 25, p-value < 2.2e-16
```

Because the **p-value $<$ 0.05**, we reject the null hypothesis of independence, and conclude that we have sufficient evidence to support there are dependence among observations in the time series. This result indicates the signal is **non-stationary**.

Approach 4: Augmented Dickey–Fuller (ADF) t-statistic test for unit root

- The ADF t-statistic test evaluates the presence of a unit root in a time series. The null hypothesis assumes the existence of a unit root, indicative of non-stationarity. A low p-value in the ADF test

suggests evidence against the null hypothesis, implying that the time series is likely stationary, without a unit root. In summary, the ADF test helps determine whether a time series is stationary or exhibits a unit root.

```
options(warn = -1)
adf.test(training)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  training
## Dickey-Fuller = -0.17584, Lag order = 3, p-value = 0.99
## alternative hypothesis: stationary
```

Because the **p-value $>$ 0.05**, we fail to reject the null hypothesis of non-stationarity, and conclude that we have no sufficient evidence to support the absence of a unit root in the time series. This result indicates that the time series is **non-stationary**.

Approach 5: Kwiatkowski-Phillips-Schmidt-Shin (KPSS) for level or trend stationarity

- The KPSS test assesses the level or trend stationarity in a time series. The null hypothesis assumes the presence of a unit root, indicating non-stationarity in the form of a trending or non-constant mean. A high p-value in the KPSS test provides evidence against the null hypothesis, suggesting that the time series is likely stationary in terms of level or trend. In summary, the KPSS test helps determine whether a time series is stationary or exhibits a non-constant mean or trend.

```
kpss.test(training, null = "Trend")
```

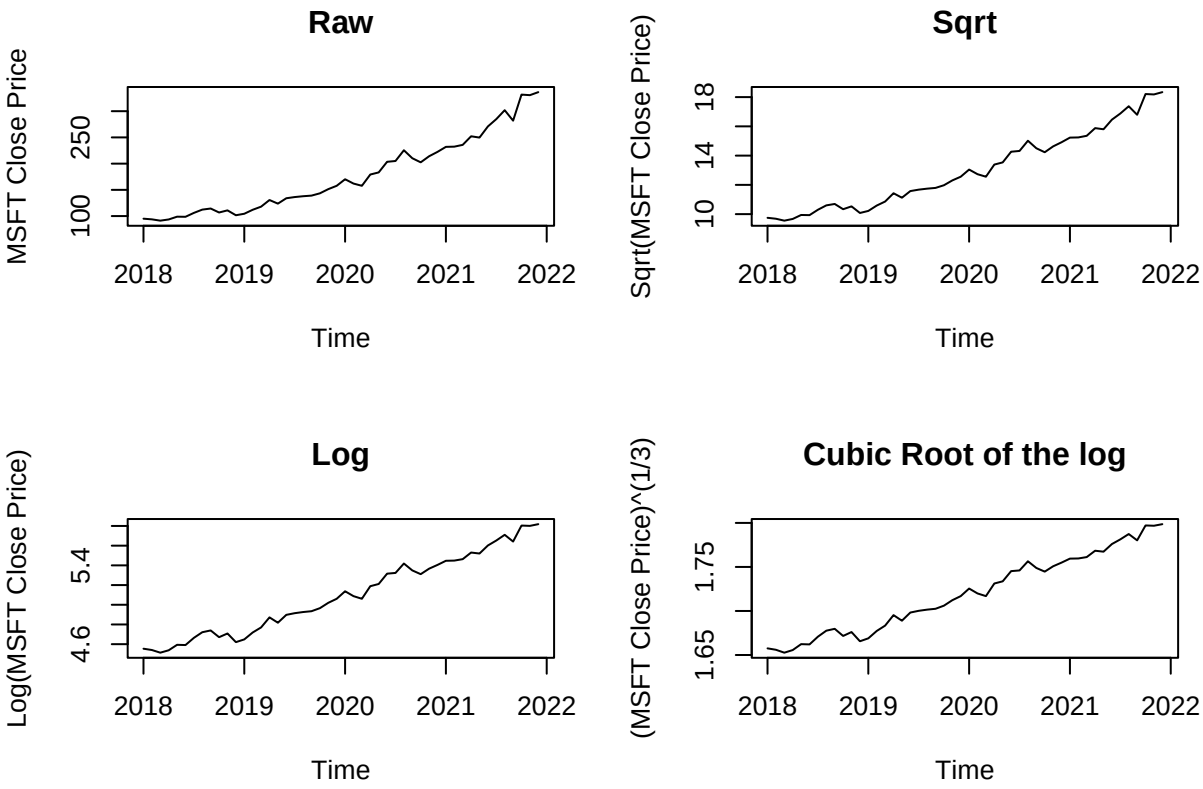
```
##
##  KPSS Test for Trend Stationarity
##
## data:  training
## KPSS Trend = 0.28919, Truncation lag parameter = 3, p-value = 0.01
```

Because the **p-value $<$ 0.05**, we reject the null hypothesis of the presence of a unit root, and conclude that we have sufficient evidence to support there is signal is not trend stationary in the time series. This result indicates the signal is **non-stationary**.

Step 2: Stationarize series (If Non-Stationary)

Address the need for pre-transformation as it sometimes useful to stabilize when variance when it is not constant over time.

```
par(mfrow = c(2,2))
plot(training, main = "Raw", ylab = "MSFT Close Price" )
plot(sqrt(training), main = "Sqrt",  ylab = "Sqrt(MSFT Close Price)")
plot(log(training), main = "Log",  ylab = "Log(MSFT Close Price)")
plot(log(training)^(1/3), main = "Cubic Root of the log",  ylab = "(MSFT Close Price)^(1/3)")
```



Sincere there is no noticeable improvement by applying the pre-transformations. (Note: Box-Cox transformation can be also included in this step)

Then, apply differencing techniques to remove the trend and/or seasonality to make the series stationary.

Regular (First) Differencing:

$$\begin{aligned}\nabla y_t &= y_t - y_{t-1} \\ &= y_t - By_t \\ &= (1-B)y_t\end{aligned}$$

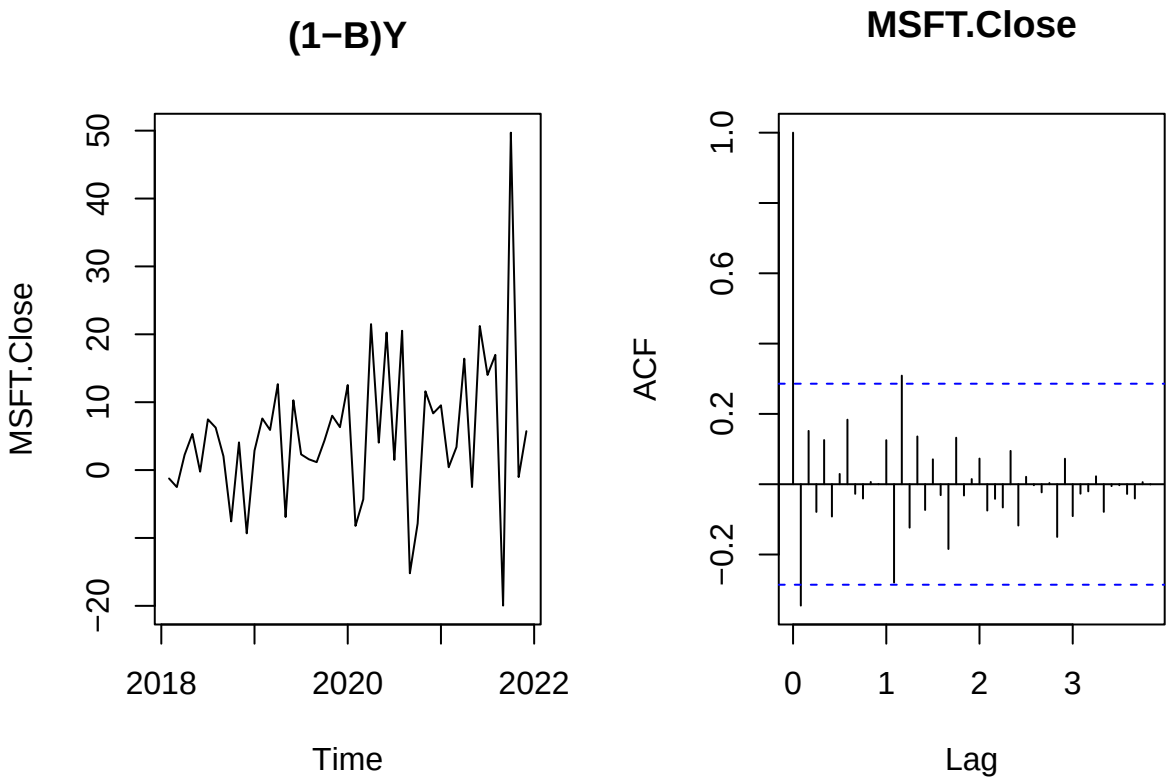
Where:

- $\nabla = (1-B)$.
- y_t denotes the value of the time series Y at period t.
- B denotes the backshif operator.

$By_t = y_{t-1}$ denotes operating on y_t with B results in a backward shift of the data by one period.

Regular differencing once usually leaves only the seasonal and the irregular part of a time series. If trend still persist, we regular difference again the difference. However, it is not common, and should be avoided as regular difference more than once result in losing a lot of data.

```
par(mfrow = c(1,2))
Reg_Diff <- diff(training, differences = 1, lag = 1)
plot(Reg_Diff, main = "(1-B)Y")
acf(Reg_Diff, lag.max = 100)
```



Seasonal Differencing:

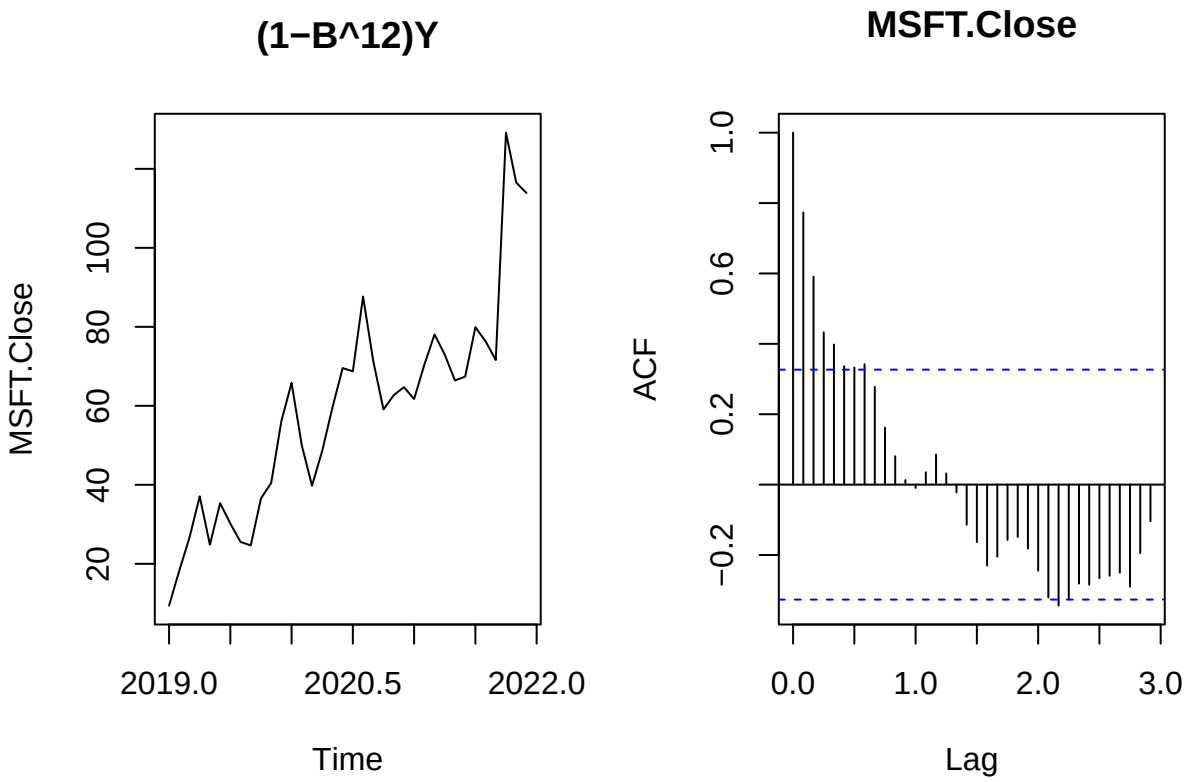
$$\begin{aligned}\nabla_m y_t &= y_t - y_{t-m} \\ &= y_t - B^m y_t \\ &= (1 - B^m) y_t\end{aligned}$$

Where:

- $\nabla = (1 - B^m)$.
- m denotes the number of observations per year.

Seasonal differencing isolates the trend and random elements, potentially including a random seasonal component and other cyclical factors. Additionally, it may address the trend in the mean.

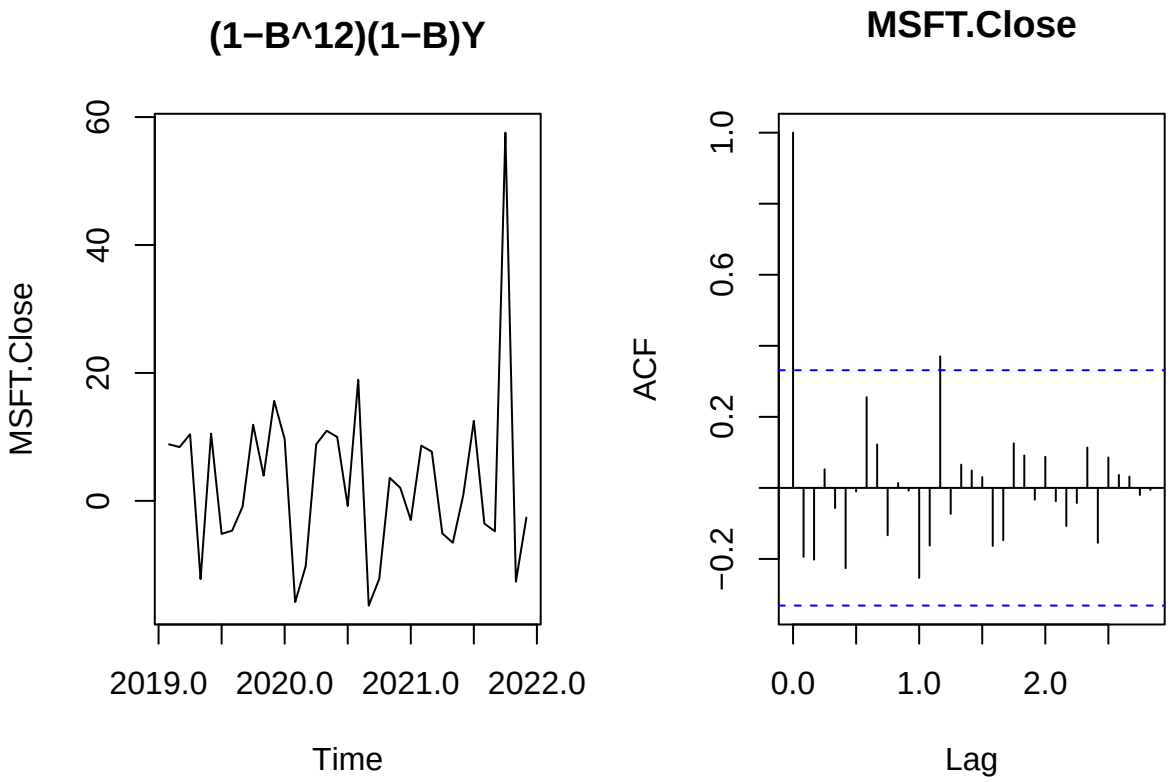
```
par(mfrow = c(1,2))
Seas_Diff <- diff(training, differences = 1, lag = 12)
plot(Seas_Diff, main = "(1-B^12)Y")
acf(Seas_Diff, lag.max = 100)
```



Seasonal Differencing of Regular Differencing:

$$\begin{aligned}\nabla_m \nabla y_t &= \nabla_m (1-B) y_t \\ &= (1-B^m)(1-B) y_t \\ &= (1 - B^m - B + B^{m+1}) y_t\end{aligned}$$

```
par(mfrow = c(1,2))
Seas_of_Reg_Diff <- diff(Reg_Diff, differences = 1, lag = 12)
plot(Seas_of_Reg_Diff, main = "(1-B^12)(1-B)Y")
acf(Seas_of_Reg_Diff, lag.max = 100)
```



Seasonal Differencing of Regular Differencing combines the benefits of both regular and seasonal differencing. However, we usually want our model as simple as possible if simple model is efficient enough.

Observing there are no clear trend and seasonal patterns in the **Regular Differecing** and the **Seasonal Differencing of Regular Differencing** time plots. In addiction, their ACF display the lags die out quickly after lags increasing. These results indicate these two differencing types are enough to make the mean of the series data stationary.

```
Box.test(Reg_Diff, lag = 25, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  Reg_Diff
## X-squared = 32.936, df = 25, p-value = 0.1327
```

```
Box.test(Seas_of_Reg_Diff, lag = 25, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  Seas_of_Reg_Diff
## X-squared = 32.127, df = 25, p-value = 0.1544
```

Both **p-value $>$ 0.05**, we fail to reject the null hypothesis of independence, and conclude that we don't have sufficient evidence to support there are dependence among observations in both differenced time series data.

Step 3: Identify Parameters (Manual Selection):

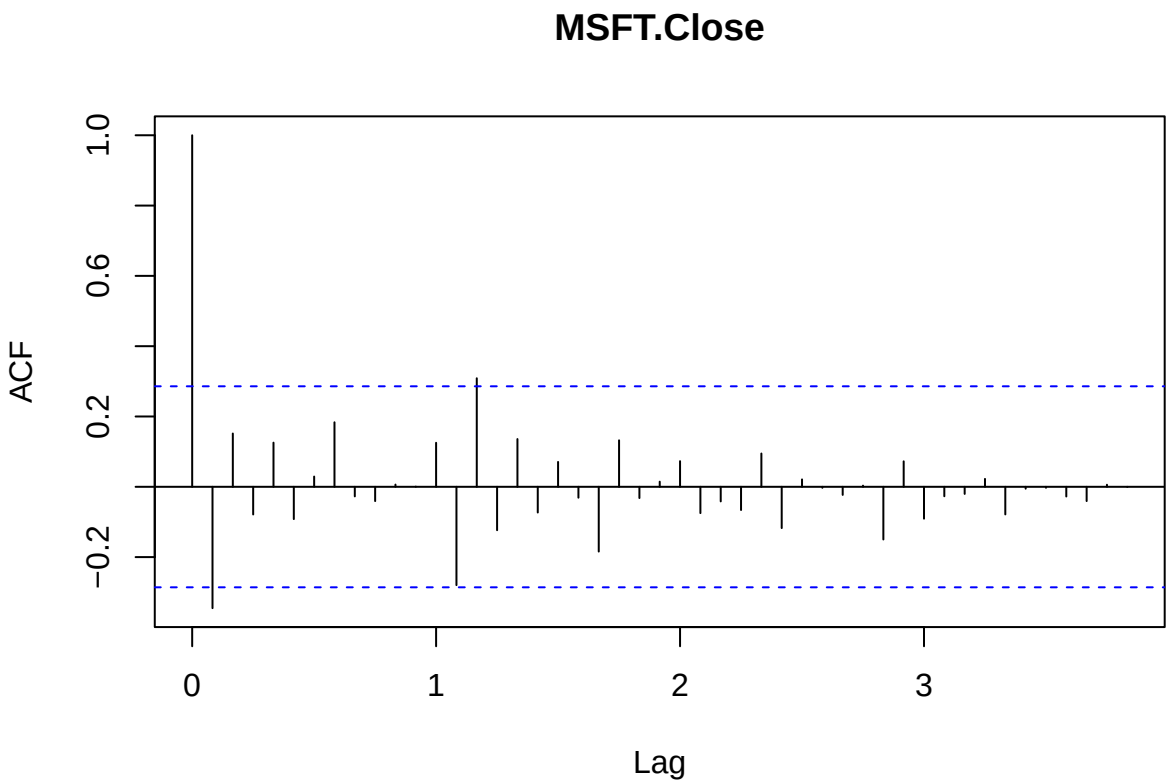
- Autocorrelation Function (ACF) measures the correlation between a time series and its own lagged values.
  - Purpose: It is used to identify the presence of autocorrelation in a time series at different lags.

- Partial AutoCorrelation Function (PACF) measures the correlation between a time series and its own lagged values, while controlling for the effect of other lags in between.
  - Purpose: It is used to identify the direct relationship between observations at different lags, excluding the influence of intermediate lags.

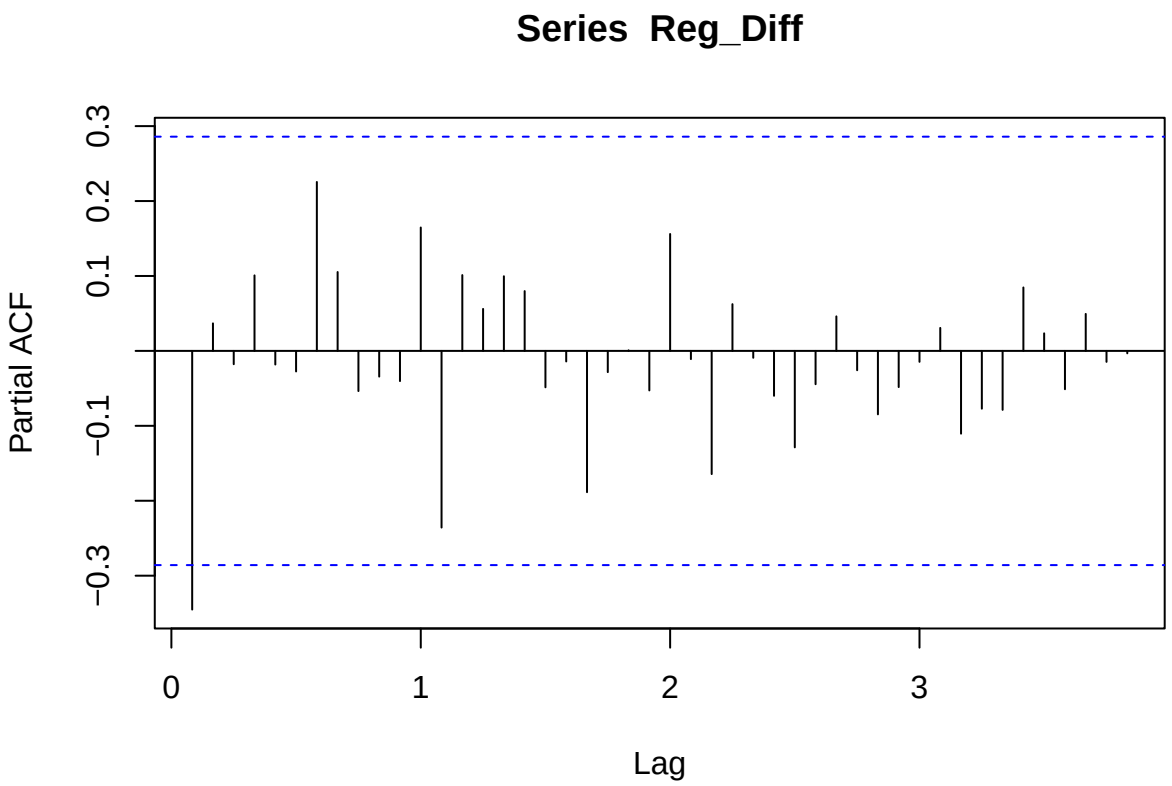
	ACF	PACF
AR(p)	Tail off (Geometric decay)	**Significant at each lag/Cuts off after lag p**
—		
MA(q)	**Significant at each lag/Cuts off after lag q**	Tail off (Geometric decay)
—		
ARMA(p, q)	Tail off (Geometric decay)	Tail off (Geometric decay)

ARIMA(1, 1, 1)(0, 0, 1)_{12}

```
acf(Reg_Diff, lag.max = 50)
```




```
pacf(Reg_Diff, lag.max = 50)
```



For **Regular Differencing:**

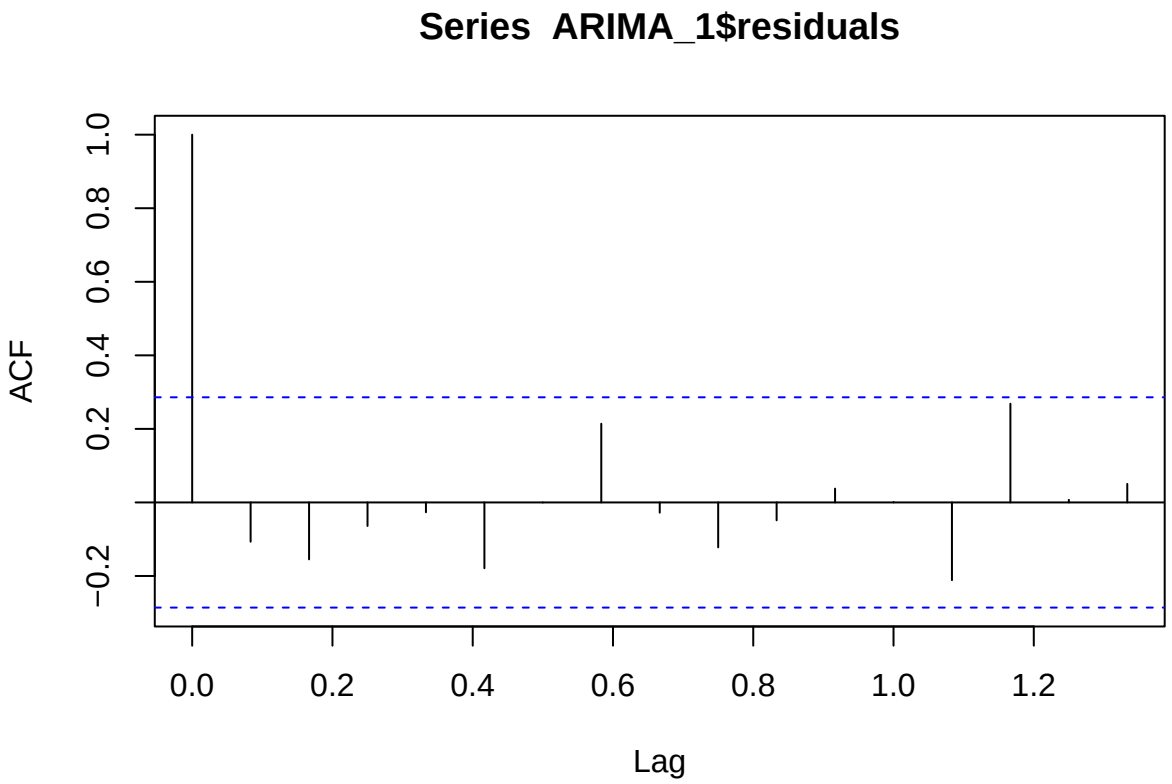
- Since both ACF and PACF are tail off, it suggests an ARMA process.
- The spikes at lags 1 and 14 in ACF plot suggests q = 1 and Q = 1.
- The spike at lag 1 in PACF plot suggests p = 1 and P = 0.
- Regular differecing once suggests d = 1 and D = 0.
- ARIMA$(1,1,1)(0,0,1)_{12}$

Note: lag 13 is similar to lag 0 which represents the correlation with itself after one seasonal period.

```
ARIMA_1 <- arima(Reg_Diff, order = c(1, 1, 1),
                 seasonal = list(order = c(0, 0, 1), period = 12))
```

Check the residual of the chosen model to ensure they are white noise:

```
acf(ARIMA_1$residuals)
```



The ACF plot of the residuals of ARIMA$(1,1,1)(0,0,1)_{12}$ model suggests white noise.

```
# Perform the absolute value of the t-statistic for coefficients
coef <- ARIMA_1$coef
se <- sqrt(diag(vcov(ARIMA_1)))
abs(coef/se)
```

```
##        ar1        ma1       sma1
##  2.7162470 13.9384261  0.3986911
```

Based on the performance of the absolute value of the t-statistic for coefficients, both ar1 and ma1 absolute t-statistics are greater than 2 suggests these two coefficients are statistically significantly different from zero; whereas the c absolute t-statistic is less than 2 suggest it is close to 0.

```
summary(ARIMA_1)
```

```
##
## Call:
## arima(x = Reg_Diff, order = c(1, 1, 1), seasonal = list(order = c(0, 0, 1),
##     period = 12))
##
## Coefficients:
##           ar1      ma1    sma1
##       -0.3871  -0.8954  0.0640
## s.e.   0.1425   0.0642  0.1605
##
## sigma^2 estimated as 110.7:  log likelihood = -174.73,  aic = 357.45
##
## Training set error measures:
##                     ME     RMSE      MAE     MPE    MAPE      MASE       ACF1
## Training set 2.194235 10.40935 7.923983 36.5105 120.794 0.6292458 -0.1067999
```

Summary: aic = 357.45, RMSE = 10.40935

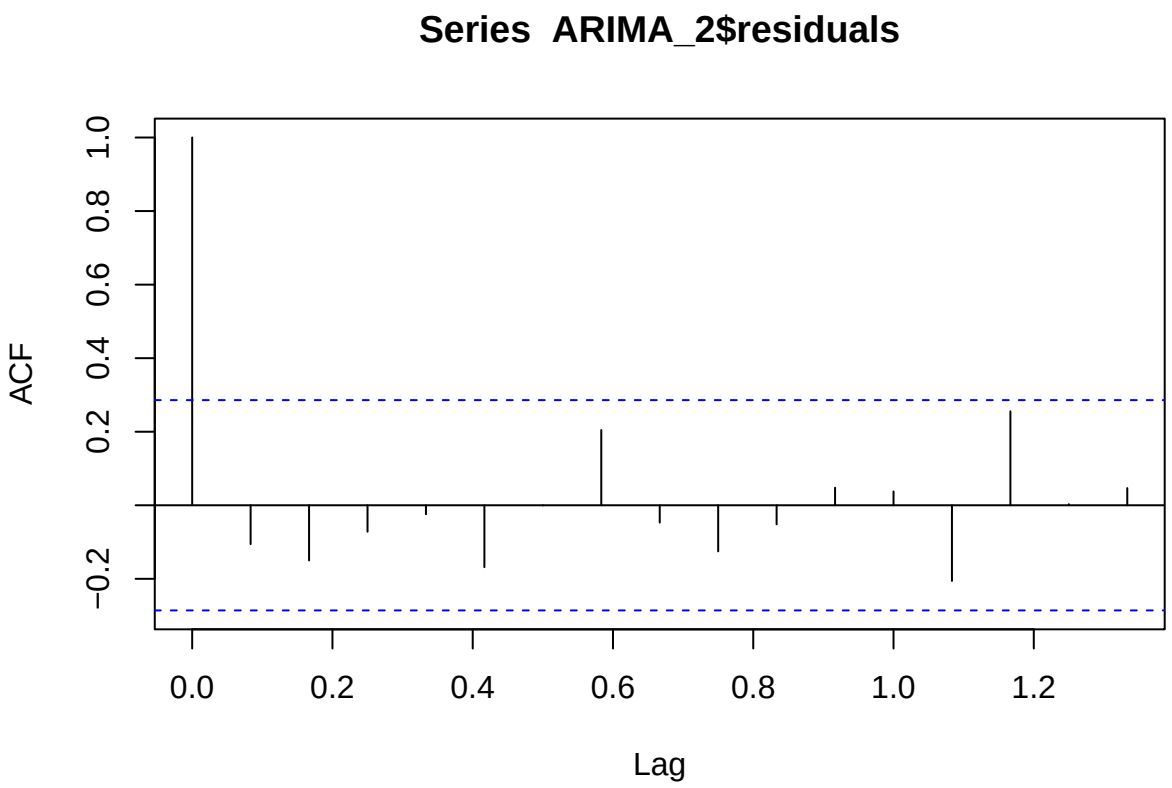
ARIMA(1, 1, 1)

Get rid of sma1 coefficient based on the T-test of ARIMA$(1,1,1)(0,0,1)_{12}$ model.

```
ARIMA_2 <- arima(Reg_Diff, order = c(1, 1, 1)) # same as
# arima(Reg_Diff, order = c(1, 1, 1),
#       seasonal = list(order = c(0, 0, 0), period = 12))
```

Check the residual of the chosen model to ensure they are white noise:

```
acf(ARIMA_2$residuals)
```



The ACF plot of the residuals of ARIMA$(1,1,1)$ model suggests white noise.

```
# Perform the absolute value of the t-statistic for coefficients
coef <- ARIMA_2$coef
se <- sqrt(diag(vcov(ARIMA_2)))
abs(coef/se)
```

```
##       ar1       ma1
##  2.931656 13.803625
```

Based on the performance of the absolute value of the t-statistic for coefficients, both ar1 and ma1 absolute t-statistics are greater than 2 suggests these two coefficients are statistically significantly different from zero.

```
summary(ARIMA_2)
```

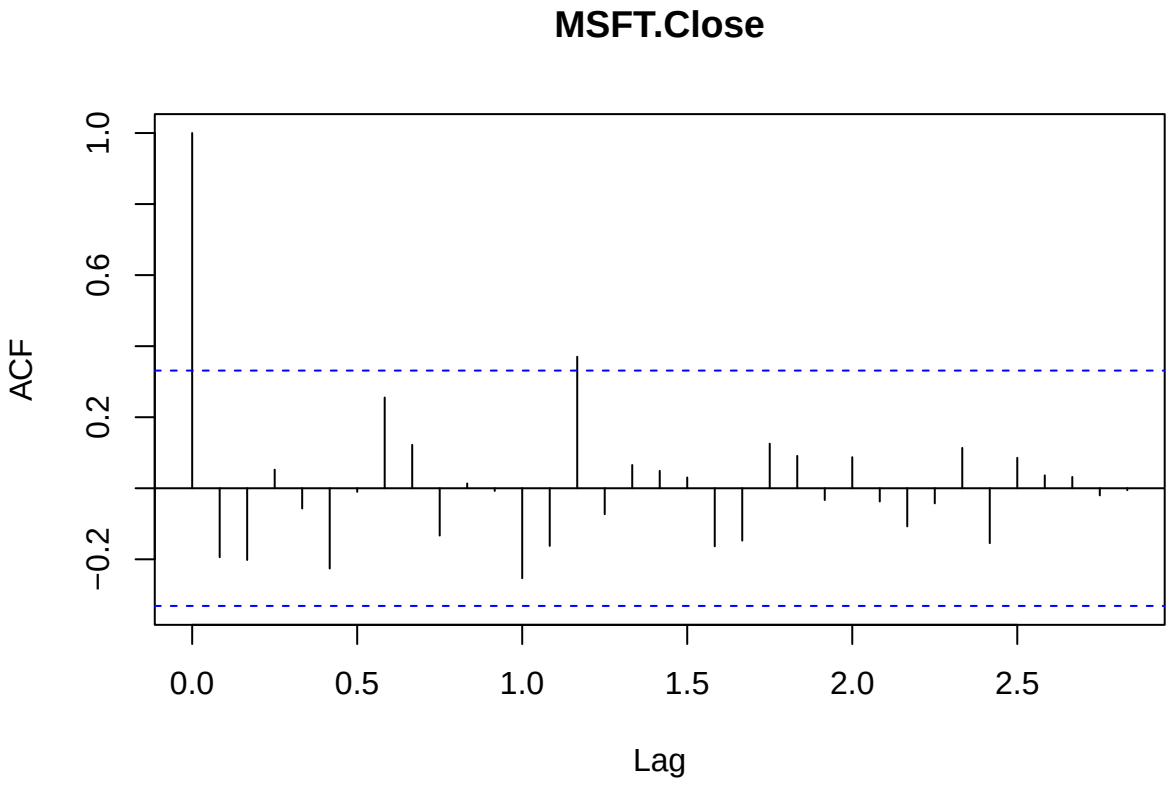
```
##
## Call:
## arima(x = Reg_Diff, order = c(1, 1, 1))
##
```

```
## Coefficients:
##           ar1      ma1
##       -0.4010  -0.8917
## s.e.   0.1368   0.0646
##
## sigma^2 estimated as 111.1:  log likelihood = -174.81,  aic = 355.61
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE       ACF1
## Training set 2.213606 10.42952 7.993148 31.16367 125.9218 0.6347382 -0.1055606
```

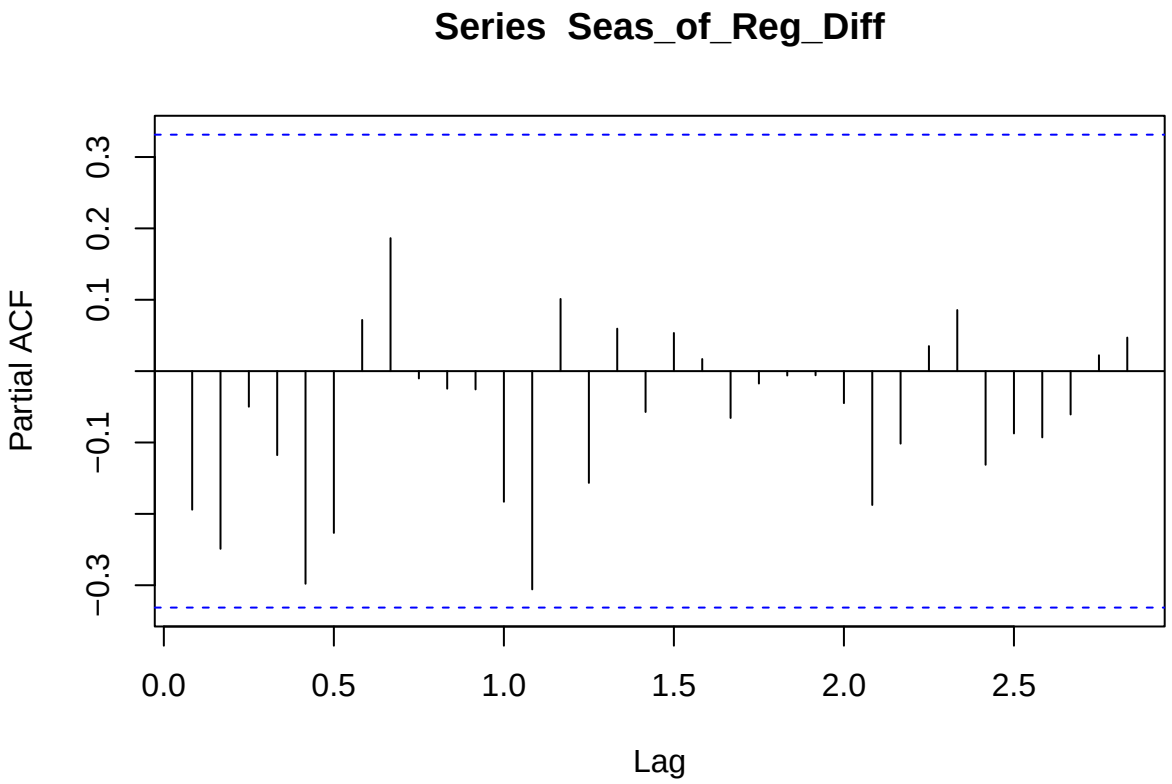
Summary: aic = 355.61, RMSE = 10.42952

ARIMA(0, 1, 0)(0, 1, 1)_{12}

```
acf(Seas_of_Reg_Diff, lag.max = 50)
```




```
pacf(Seas_of_Reg_Diff, lag.max = 50)
```



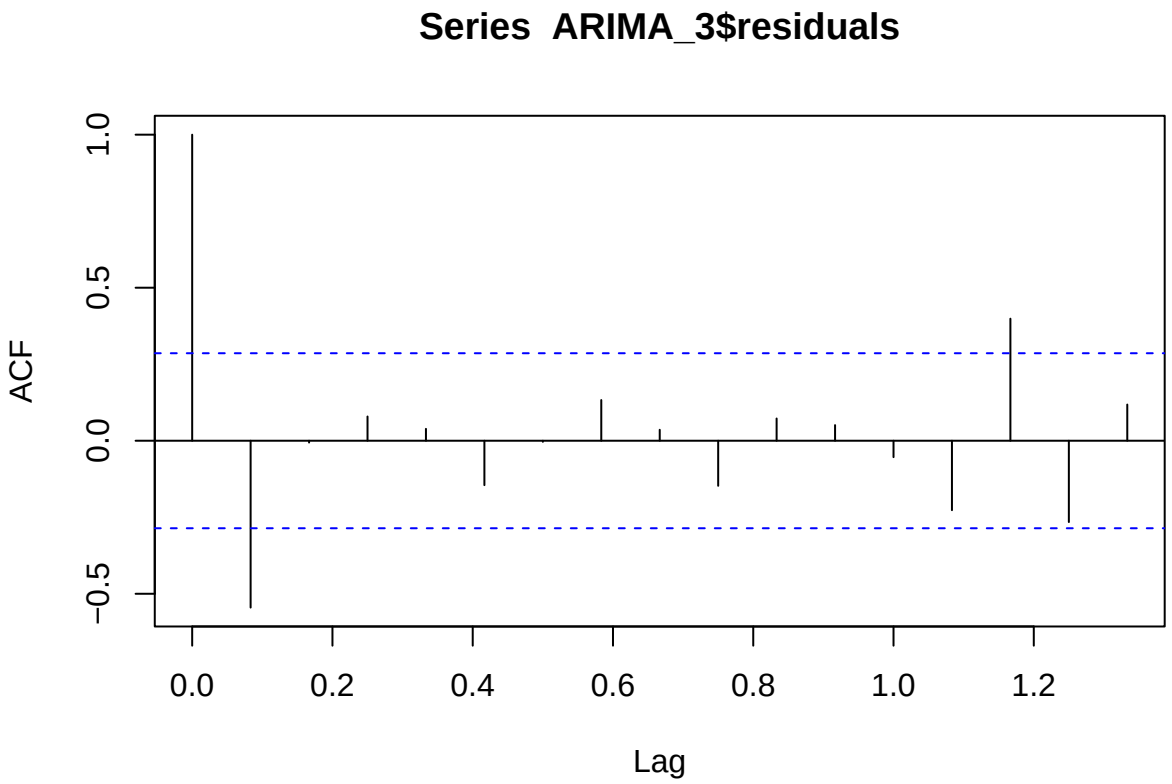
For **Seasonal Differencing of Regular Differencing:**

- Since both ACF and PACF are tail off, it suggests an ARMA process.
- The spikes at lag 14 in ACF plot suggests q = 0 and Q = 1.
- There is no spikes in PACF plot suggests p = 0 and P = 0.
- Regular differecing once suggests d = 1 and D = 1.
- ARIMA$(0,1,0)(0,1,1)_{12}$

```
ARIMA_3 <- arima(Reg_Diff, order = c(0, 1, 0),
                 seasonal = list(order = c(0, 1, 1), period = 12))
```

Check the residual of the chosen model to ensure they are white noise:

```
acf(ARIMA_3$residuals)
```



The ACF plot of the residuals of ARIMA$(0,1,0)(0,1,1)_{12}$ model is not white noise. Repeat **Step 3** to choose better parameters for ARIMA$(p,d,q)(P,D,Q)_m$.

Summary

Based on the ***summary table:***

- ARIMA$(1,1,1)(0,0,1)_{12}$: ***aic = 357.45, RMSE = 10.40935***
- ARIMA$(1,1,1)$: ***aic = 355.61, RMSE = 10.42952***

The lower Akaike Information Criteria (AIC), the better quality and goodness-of-fit to the data. ARIMA$(1,1,1)$ model has lower aic value, and it seems perform better. However, both models have high RMSE values, indicating that the average magnitude of the errors between predicted and actual values is large. It concluded that both predictive models are expected not performing well.

Forecast

12 months predictions for ARIMA$(1,1,1)(0,0,1)_{12}$ and ARIMA$(1,1,1)$ models.

```
ARIMA_1_fcast <- predict(ARIMA_1, n.ahead = 12)
ARIMA_2_fcast <- predict(ARIMA_2, n.ahead = 12)

ARIMA_1_fcast$pred
```

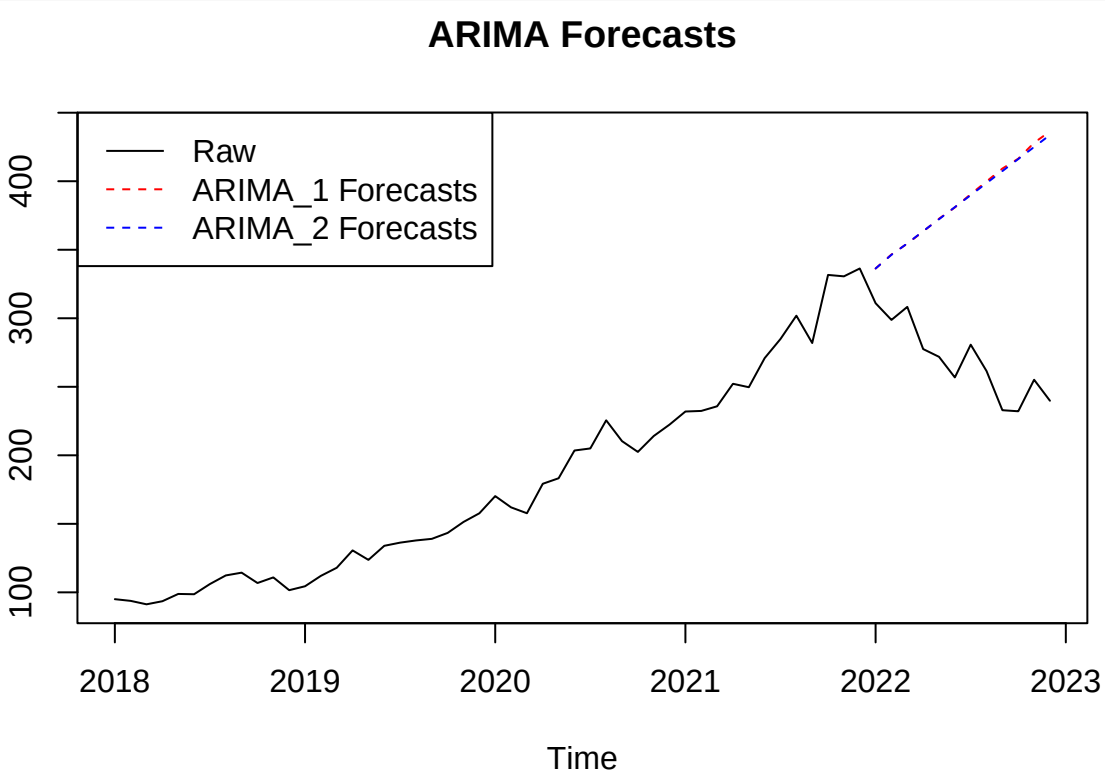
```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2022 10.193505  7.988979  8.823507  9.296878  8.252105  9.669907  9.297959
##            Aug       Sep       Oct       Nov       Dec
## 2022  9.403573  7.188683 11.616701  8.291596  8.737170
```

Observing the predicted values are far from the actual values. It is because the predicted values are obtained from the differenced model which is working with the changes between consecutive values of subtracting each

value from its previous value. Therefore, the inverse differencing operation is needed to revert the forecasted differences back to the original scale for predicting the raw price.

```
# Inverse differencing
ARIMA_1_fcast$pred <- cumsum(c(training[length(training)], ARIMA_1_fcast$pred))
ARIMA_2_fcast$pred <- cumsum(c(training[length(training)], ARIMA_2_fcast$pred))
```

```
ARIMA_1_forecasts <- ts(ARIMA_1_fcast$pred,
                        start = c(2022, 1), end = c(2022, 12),
                        frequency = 12)
ARIMA_2_forecasts <- ts(ARIMA_2_fcast$pred,
                        start = c(2022, 1), end = c(2022, 12),
                        frequency = 12)

ts.plot(cbind(ts, ARIMA_1_forecasts, ARIMA_2_forecasts),
        lty = c(1, 2, 2),
        col = c("black", "red", "blue"),
        main = "ARIMA Forecasts")
legend("topleft",
       legend = c("Raw", "ARIMA_1 Forecasts", "ARIMA_2 Forecasts"),
       col = c("black", "red", "blue"),
       lty = c(1, 2, 2))
```



This visualization results is consistent with the summary findings.

Step 3: Auto.ARIMA (Automated Algorithm):

Using the **Regular Differecing** and the **Seasonal Differencing of Regular Differencing** time series data from **Step 2**.

```
auto_ARIMA_1 <- auto.arima(Reg_Diff)
summary(auto_ARIMA_1)
```

```
## Series: Reg_Diff
## ARIMA(0,1,2)
##
## Coefficients:
##           ma1     ma2
##       -1.3475  0.4349
## s.e.   0.1448  0.1395
##
## sigma^2 = 114.1:  log likelihood = -174.49
## AIC=354.98   AICc=355.55   BIC=360.46
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
## Training set 2.27593 10.33749 7.745118 39.02785 115.5117 0.8031499 -0.07971295
```

Summary: $\boldsymbol{ARIMA(0,1,2)}$***, aic = 354.98, RMSE = 10.33749***

```
auto_ARIMA_2 <- auto.arima(Seas_of_Reg_Diff)
summary(auto_ARIMA_2)
```

```
## Series: Seas_of_Reg_Diff
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 182.6:  log likelihood = -140.79
## AIC=283.58   AICc=283.7   BIC=285.13
##
## Training set error measures:
##                    ME     RMSE      MAE MPE MAPE      MASE       ACF1
## Training set 2.985143 13.51256 9.643427 100  100 0.5782336 -0.1941081
```

Summary: $\boldsymbol{ARIMA(0,0,0)}$***, aic = 283.58, RMSE = 13.51256***

```
auto_ARIMA_3 <- auto.arima(training)
summary(auto_ARIMA_3)
```

```
## Series: training
## ARIMA(0,2,2)
##
## Coefficients:
##           ma1     ma2
##       -1.3475  0.4349
## s.e.   0.1448  0.1395
##
## sigma^2 = 114.2:  log likelihood = -174.49
## AIC=354.98   AICc=355.55   BIC=360.46
##
## Training set error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
## Training set 2.226694 10.22925 7.587332 1.061219 4.233935 0.1290216 -0.07836279
```

Summary: $\boldsymbol{ARIMA(0,2,2)}$***, aic = 354.98, RMSE = 10.22925***

Summary

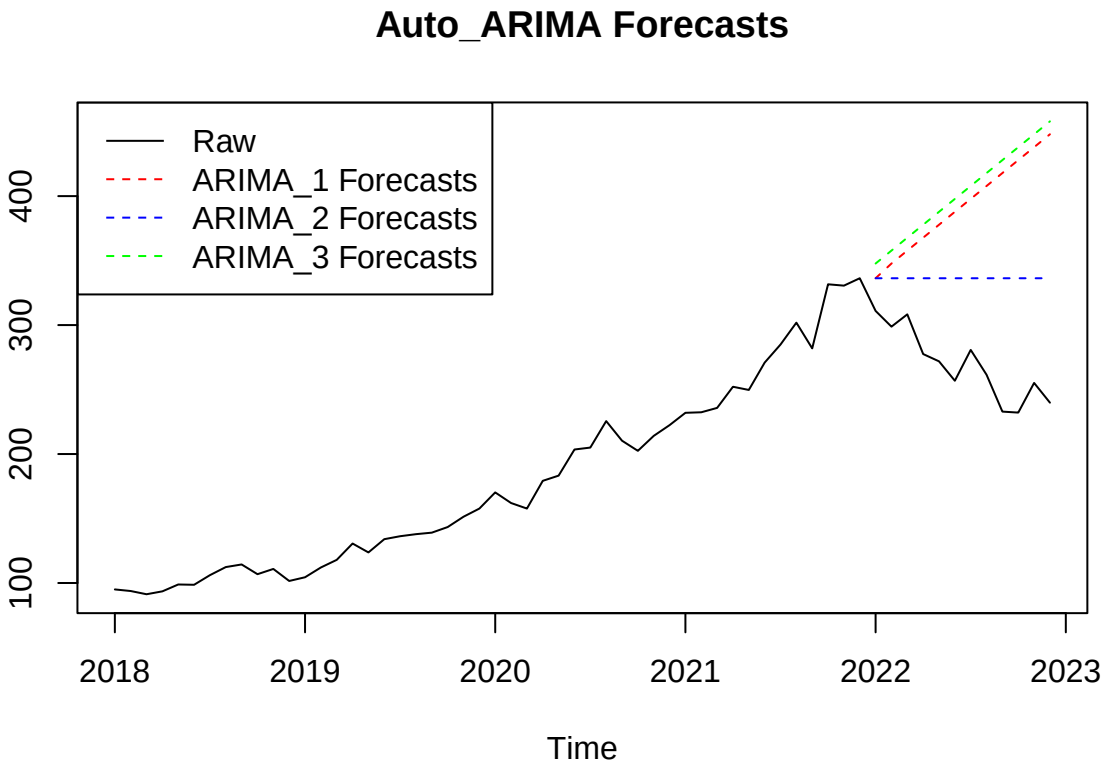
- ARIMA$(0,1,2)$: ***aic = 354.98, RMSE = 10.33749***
- ARIMA$(0,0,0)$: ***aic = 283.58, RMSE = 13.51256***
- ARIMA$(0,2,2)$: ***aic = 354.98, RMSE = 10.22925***

Based on the ***summary table:***

It results the same with manually select the parameters. ARIAM models selected by automated algorithm also with high RMSE values, resulting the same interpretation with the manually selected models.

Forecast

```
auto_ARIMA_1_fcast <- predict(auto_ARIMA_1, n.ahead = 12)
auto_ARIMA_2_fcast <- predict(auto_ARIMA_2, n.ahead = 12)
auto_ARIMA_3_fcast <- predict(auto_ARIMA_3, n.ahead = 12)

auto_ARIMA_1_fcast$pred <- cumsum(c(training[length(training)], auto_ARIMA_1_fcast$pred))
auto_ARIMA_2_fcast$pred <- cumsum(c(training[length(training)], auto_ARIMA_2_fcast$pred))
# there is no need for inverse differencing for training data

auto_ARIMA_1_forecasts <- ts(auto_ARIMA_1_fcast$pred,
                             start = c(2022, 1), end = c(2022, 12),
                             frequency = 12)
auto_ARIMA_2_forecasts <- ts(auto_ARIMA_2_fcast$pred,
                             start = c(2022, 1), end = c(2022, 12),
                             frequency = 12)
auto_ARIMA_3_forecasts <- ts(auto_ARIMA_3_fcast$pred,
                             start = c(2022, 1), end = c(2022, 12),
                             frequency = 12)

ts.plot(cbind(ts, auto_ARIMA_1_forecasts, auto_ARIMA_2_forecasts, auto_ARIMA_3_forecasts),
        lty = c(1, 2, 2, 2),
        col = c("black", "red", "blue", "green"),
        main = "Auto_ARIMA Forecasts")
legend("topleft",
       legend = c("Raw", "ARIMA_1 Forecasts", "ARIMA_2 Forecasts", "ARIMA_3 Forecasts"),
       col = c("black", "red", "blue", "green"),
       lty = c(1, 2, 2, 2))
```



This visualization results is consistent with the summary findings.

Conclusion

Although the ARIMA model can address non-stationarity through its integrated (**I**) component and offer interpretable outcomes, its forecasted results may fall short in capturing sudden changes, particularly when dealing with complex or nonlinear patterns in financial markets.

Limitations & Recommendations

- 1. ARIMA models (manual selection of parameters) can lead to less accurate predictions from the optimal result.
  - Having a domain expertise and knowledge of specific time series data can capture some reality patterns, which potentially perform better than the automated selected optimal parameters by the automated algorithm.
- 2. ARIMA models face limitations in capturing complex or non-linear patterns in financial markets due to their inherent linearity. This constraint poses a challenge for the ability of model to effectively comprehend sudden changes or rare events that could significantly impact asset prices.
  - Aiming to transform input parameters and make them more suitable to linear modeling with some feature engineering techniques.
- 3. ARIMA models are dependent on high-quality data and are built on assumptions such as stationarity and normality. Violating these assumptions can compromise the accuracy and reliability of the model.
  - Considering alternative modeling such as GARCH or other machine learning models which are less sensitive to assumptions of stationarity and normality
- 4. ARIMA models primarily rely on historical price data and may not readily integrate external factors, such as news events, economic indicators, or market sentiment. These factors are crucial

in influencing price movements but may not be incorporated into the framework.

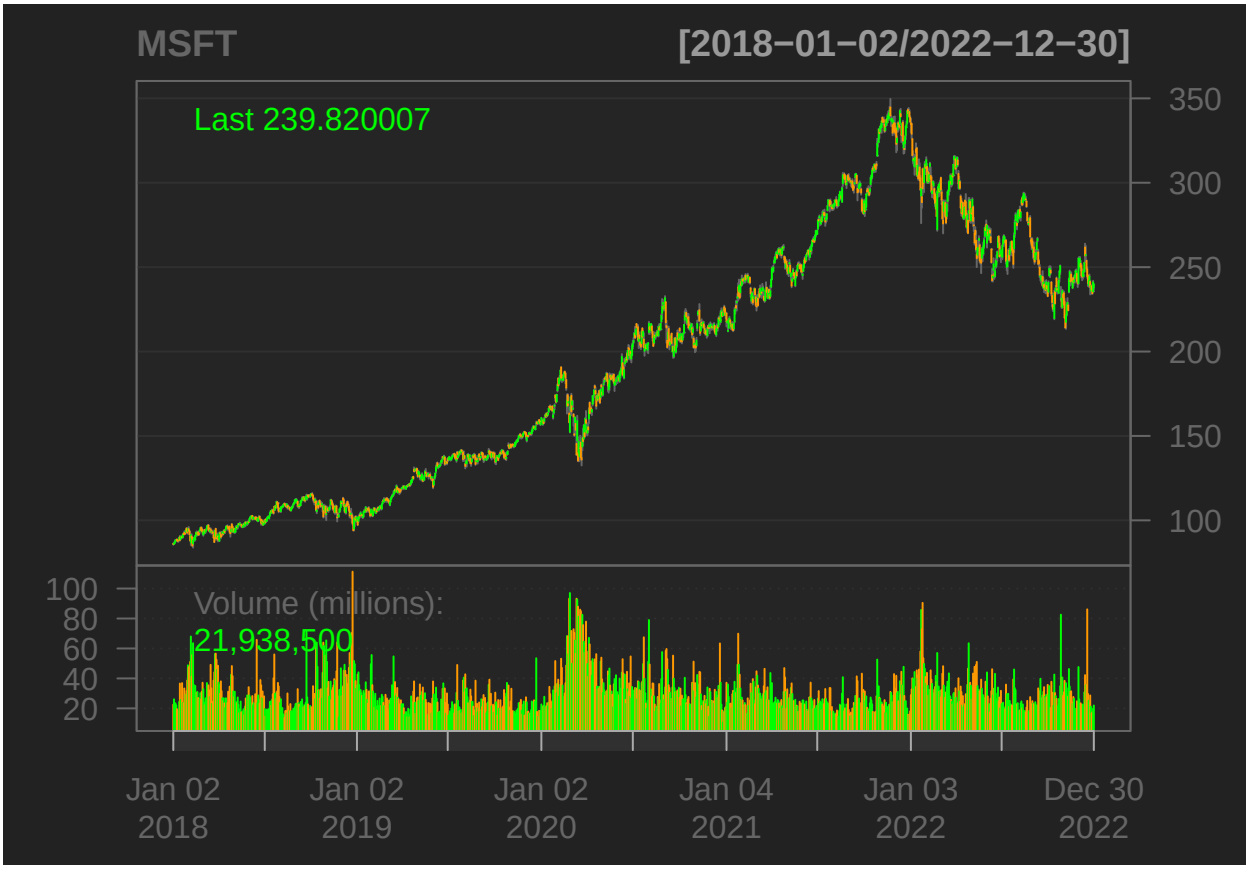
– Considering alternative advanced models such as hybrid models that combine time series analysis with machine learning techniques, allowing for better incorporation of external factors.

Generalized AutoRegressive Conditional Heteroscedasticity (GARCH)

- Application: GARCH is used in analyzing time-series data where the variance error is believed to be serially autocorrelated under the assumption that the variance of the error term follows an autoregressive moving average process. GARCH is widely used to help predict the volatility of returns on financial assets, and to assess risk and expected returns for assets that exhibit clustered periods of volatility in returns.

In this study, it will be used to forecast the volatility, and help to understand other basic concepts, including few types of GARCH models such as SGARCH and gjrGARCH.

```
chartSeries(MSFT)
```



It is important to calculate the returns before modeling volatility with GARCH model for capturing the conditional heteroskedastic nature of financial time series data.

- Conditional: the volatility of the series is not fixed over time, it's based on where you at.
- Heteroscedasticity: the variances do not remain same over time.
- Volatility: the degree of variation of a trading price series over a certain period of time. The higher volatility, the larger prices swings, and vice versa.

So, we want to convert into the daily returns data to normalize the data so that we can do comparisons

between different assets or investments regardless the price levels.

```
returns <- CalculateReturns(MSFT$MSFT.Close)
returns <- returns[-1]
```

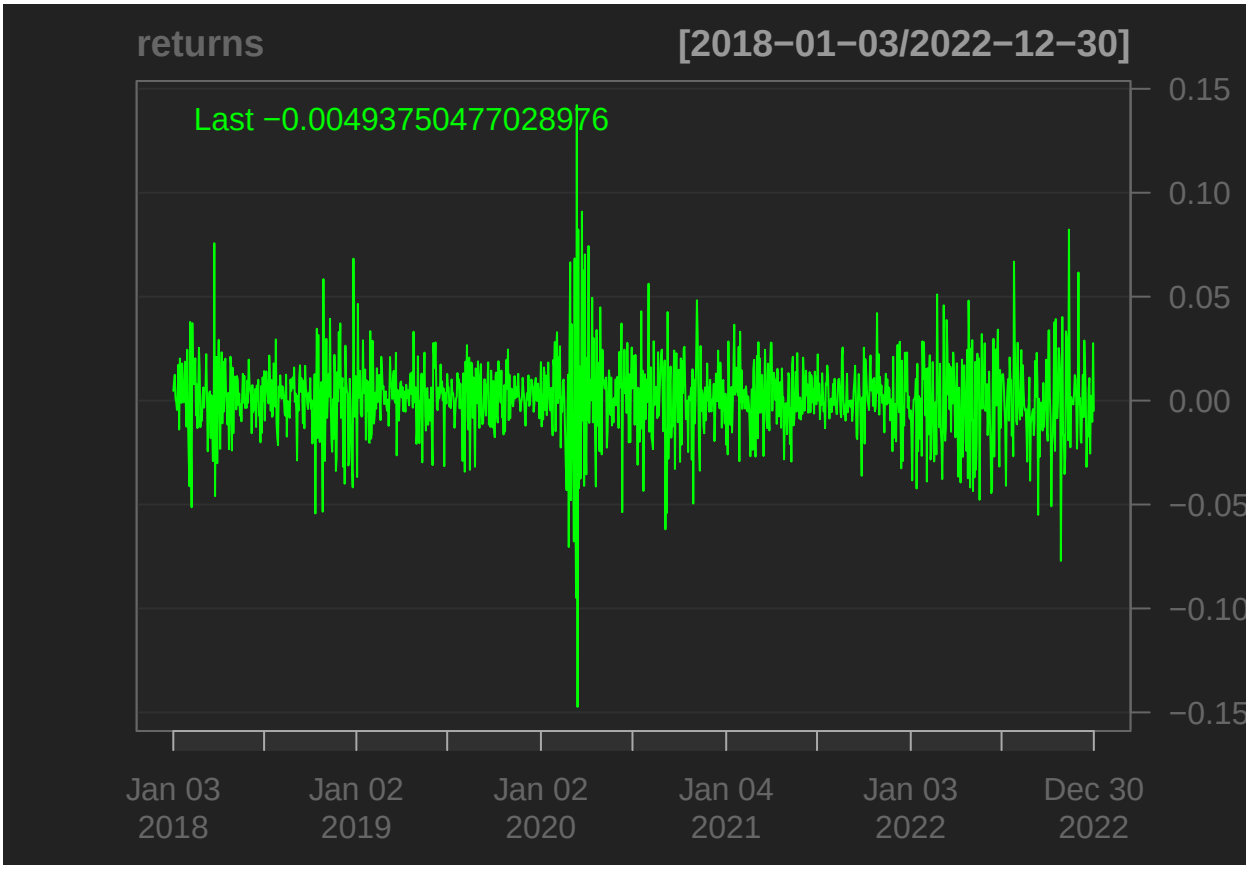
Relative financial gains and losses, expressed in terms of return:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

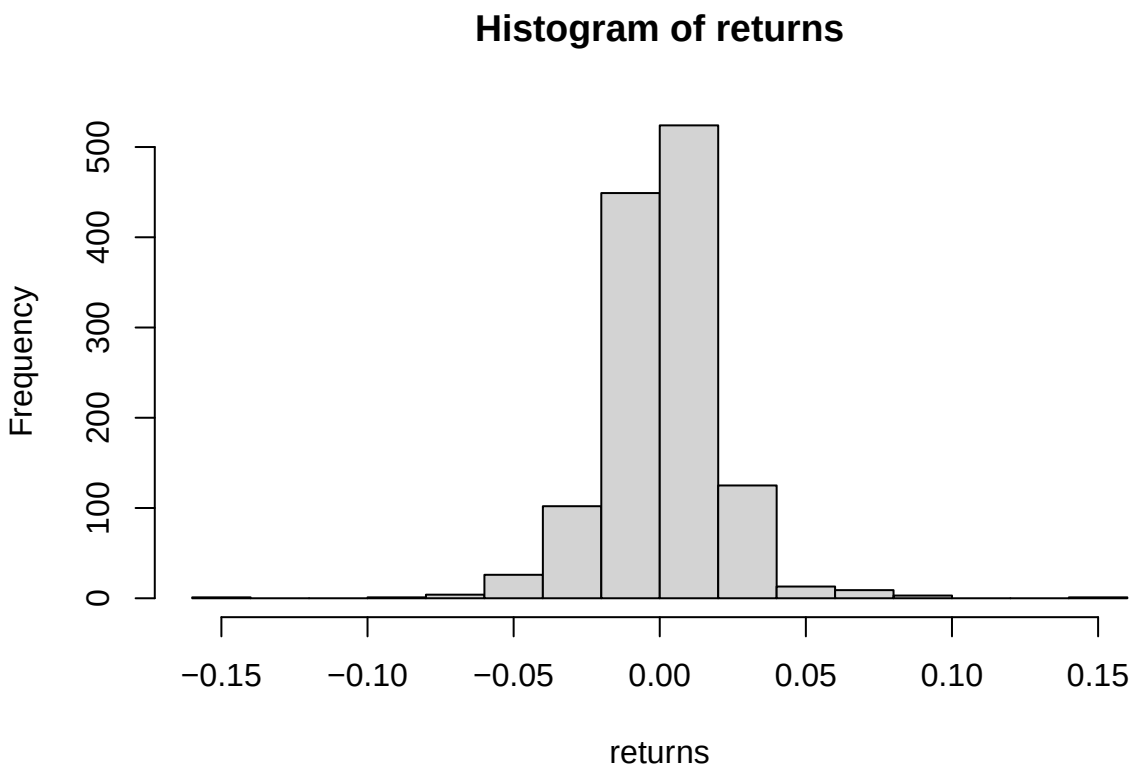
Properties of daily returns:

- The average return is zero
- Return variability changes through time

```
chartSeries(returns)
```




```
hist(returns)
```



It provides a standardized way to assess the relative movement of asset prices on a daily basis.

The estimated return volatility:

$$\hat{\sigma} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(R_t - \hat{\mu})^2}$$

Where:

- T denotes the daily returns.
- $\hat{\mu}$ denotes the mean return.

```
sd(returns) # measure of return variability
```

```
## [1] 0.01956364
```

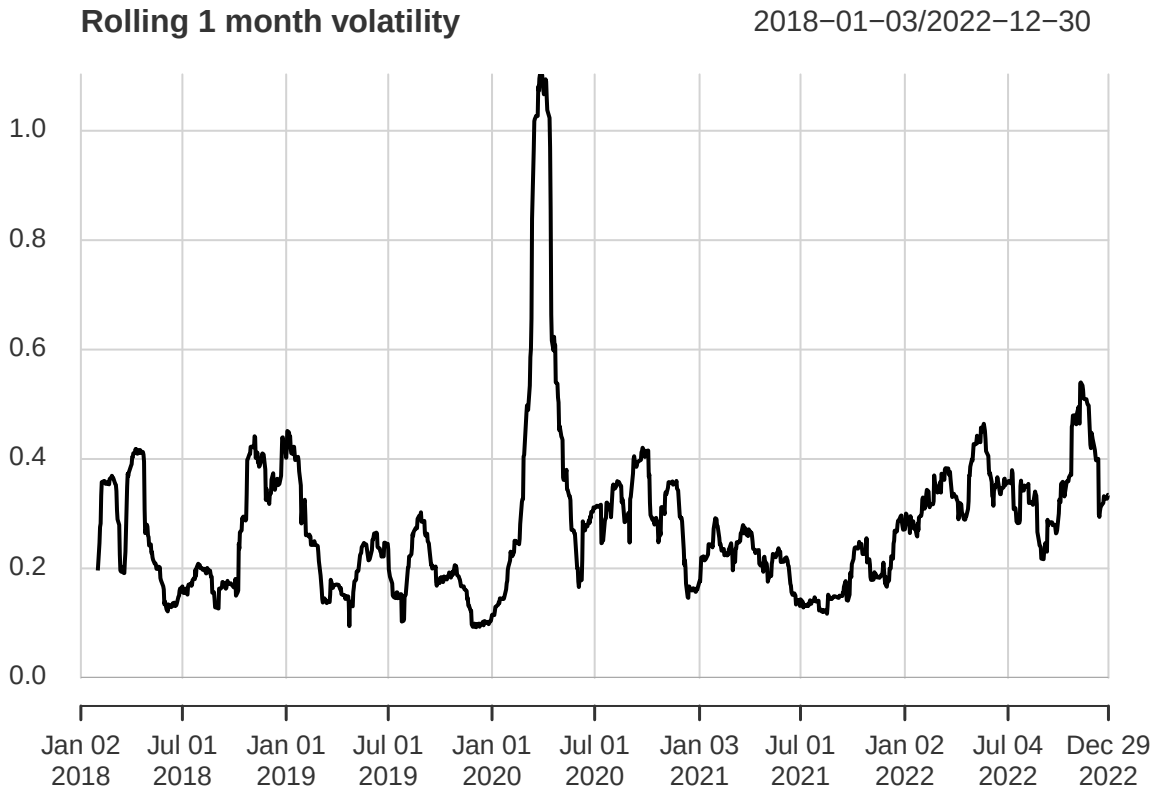
```
sqrt(252) * sd(returns) # annualized volatility
```

```
## [1] 0.3105632
```

It represents the annualized (252 trading days) volatility of the returns.

Roll the sample through time by adding the most recent observation and removing the most distant one. It tells the volatility has been in the past.

```
chart.RollingPerformance(R = returns,
                         width = 22, # trading days in a month
                         FUN = "sd.annualized",
                         scale = 252,
                         main = "Rolling 1 month volatility")
```



There is a noticeable high volatility in early 2020, indicating there is a huge price swings. Let's zoom into the year 2020.

```
chart.RollingPerformance(R = returns["2020-01-01/2020-12-31", ],
                         width = 22, # trading days in a month
                         FUN = "sd.annualized",
                         scale = 252,
                         main = "Rolling 1 month volatility - 2020")
```

This plot shows a notable surge in volatility from late February to April, corresponding to the onset of the pandemic. This suggests that adverse news significantly impacted Microsoft's stock prices during that period. Subsequently, from May onward, volatility stabilized, reflecting a return to a more consistent level of variance. This could be attributed to investors gaining clarity and adapting to the evolving economic landscape. In essence, the analysis highlights the profound influence of the COVID-19 pandemic on Microsoft's stock performance throughout 2020.

$$\cdots \to R_{t-3} \to R_{t-2} \to R_{t-1} \to R_t$$

To predict the future return R_t based on the information set (I_{t-1}) available at time t_1

Prediction of mean turn:

$$\mu_t = \mathrm{E}[R_t|\mathrm{I}_{t-1}]$$

Prediction error:

$$e_t = R_t - \mu_t$$

Prediction of variance:

$$\begin{aligned}\sigma_t^2 &= var(R_t|\mathrm{I}_{t-1}) \\ &= \mathrm{E}[(R_t - \mu_t)^2|\mathrm{I}_{t-1}] \\ &= \mathrm{E}[e_t^2|\mathrm{I}_{t-1}] \\ \sigma_t &= \sqrt{\sigma_t^2}\end{aligned}$$

Rolling mean model:

$$\mu_t = \frac{1}{M}\sum_{i=1}^{M} R_{t-1}$$

Rolling variance model:

$$\mu_t = \frac{1}{M} \sum_{i=1}^{M} e_{t-1}^2$$

Where:

- M denotes the most recently observed M predicted returns/errors.

ARCH

Since the future variance is more affected by the more recent events than by the distant one. For improving a forecasting accuracy by giving more weight to the most recent observation.

ARCH(p) model:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i e_{t-i}^2$$

where:

- σ_t^2 denotes the conditional variance at time t.
- α denotes the coefficient associated with the lagged squared residuals e_{t-i}^2, representing the impact of past unexpected return (shocks) on the current variance.
- ω denotes the constant/intercept of the model.
- p denotes the most recent observation.

ARCH(p) process is uncorrelated and has a constant mean (both conditional and unconditional) and a constant unconditional variance, but its conditional variance is nonconstant. However, it is not practical in the real life as it is not able to capture persistence in volatility over time.

GARCH(p, q) model:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2$$

where:

- α denotes the coefficient associated with the lagged squared residuals e_{t-i}^2, representing the impact of past unexpected return (shocks) on the current variance.
- β denotes the coefficient associated with the lagged conditional variance σ_{t-i}^2, representing the persistence of past conditional volatility in the current conditional variance.

GARCH model is more flexible and is able to capture both the impact of past squared errors and the persistence in volatility over time.

Parameter restrictions:

- 1. ω, α, and β are > 0: this ensures that σ_t^2 is always positive.
- 2. α and $\beta < 1$: this ensures that the predicted variance σ_t^2 always returns to the long run variance.
  - This long run variance refers to *mean-reverting*.
  - It equals to $\frac{\omega}{1-\alpha-\beta}$

The reason why the GARCH model, and its variations require a conditional distribution on returns is because financial time series data often exhibits heteroskedasticity, meaning that the volatility of the series changes over time. In other words, the variance of the series is not constant; it varies depending on past observations.

The GARCH model assumes that the conditional variance of the current observation depends on the past observations, which is expressed through autoregressive terms. The conditional distribution is necessary because it captures the relationship between the current observation's volatility and the past observations' volatility.

GARCH(1, 1) with Normal distribution

For Simplicity, assume the return is normally distributed with constant mean, variance.

$$
\begin{aligned}
R_t &= \mu + e_t \\
e_t &\sim N(0, \sigma^2) \\
\sigma_t^2 &= \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2
\end{aligned}
$$

Three steps:

- Firstly, specify the GARCH model with mean μ_t, variance σ_t^2, distribution of errors e_t.
- Secondly, estimate the GARCH model.
- Finally, predict volatility based on the estimated GARCH model.

```
garchspec_1 <- ugarchspec(mean.model = list(armaOrder = c(0, 0)),
                          variance.model = list(model = "sGARCH"),
                          distribution.model = "norm")

garchfit_1 <- ugarchfit(data = returns, spec = garchspec_1)

garchforecast_1 <- ugarchforecast(fitORspec = garchfit_1, n.ahead = 5)
```

Model Diagnosis of Checking Model Validity

```
garchfit_1
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001521    0.000400   3.8023 0.000143
## omega   0.000011    0.000003   3.6168 0.000298
## alpha1  0.158631    0.014225  11.1516 0.000000
## beta1   0.818201    0.017400  47.0234 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001521    0.000358   4.2533 0.000021
## omega   0.000011    0.000006   1.7359 0.082587
```

```
## alpha1  0.158631    0.030275   5.2396 0.000000
## beta1   0.818201    0.030154  27.1345 0.000000
##
## LogLikelihood : 3356.619
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3301
## Bayes        -5.3137
## Shibata      -5.3301
## Hannan-Quinn -5.3239
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      6.132 0.01328
## Lag[2*(p+q)+(p+q)-1][2]     6.706 0.01427
## Lag[4*(p+q)+(p+q)-1][5]     7.309 0.04360
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.03534  0.8509
## Lag[2*(p+q)+(p+q)-1][5]   0.50215  0.9573
## Lag[4*(p+q)+(p+q)-1][9]   1.98516  0.9068
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1191 0.500 2.000  0.7300
## ARCH Lag[5]    0.9608 1.440 1.667  0.7446
## ARCH Lag[7]    1.5142 2.315 1.543  0.8185
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  11.6829
## Individual Statistics:
## mu     0.1655
## omega  1.2408
## alpha1 0.1643
## beta1  0.5084
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias          0.06015 0.95204
```

```
## Negative Sign Bias 1.80756 0.07091   *
## Positive Sign Bias 0.72083 0.47115
## Joint Effect       6.46268 0.09115   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     39.55     0.003746
## 2    30     51.51     0.006183
## 3    40     65.94     0.004482
## 4    50     68.71     0.032990
##
##
## Elapsed time : 0.1664741
```

Find the optimal setting based on the estimation results

- The **Optimal Parameters** table provides the estimates, standard errors, t-values, and p-values for each coefficient. The p-values are associated with t-tests that assess whether the estimated coefficients are significantly different from zero. If the p-value is less than the significant level, reject the null hypothesis that the coefficient is equal to zero.
- The **Information Criteria** table provides Akaike (AIC), Bayes (BIC), Shibata, and Hannan-Quinn criteria for the model estimation. The lower these values, the better the model is in terms of fitting.
- The **Weighted Ljung-Box Test on Standardized Residuals** table provides the lag values for which the test is conducted. Each row corresponds to a specific lag, its test statistic, and its p-value. We want these p-values high to show there are no significant autocorrelation at tested lags, indicating a good fit.
- The **Adjusted Pearson Goodness-of-Fit Test** table provides group of error term, its test statistic, and its p-value. We want these these p-values high to show the conditional error term follows the assumed distribution.(In this case, H_o : the conditional error term follows a normal distribution.) - QQ-plot can provide the visualization.

Summary:

- 1. Based on the **Optimal Parameters** table, all parameters have p-values less than 0.05, indicating they are off from 0 and statistically significant to the model.
- 2. **LogLikelihood: 3356.619** and **AIC: -5.3301**
- 3. Based on the **Weighted Ljung-Box Test on Standardized Residuals** table, p-values of tested lags are less than 0.05, indicating there are significant autocorrelation. It suggests the model is not a good fit.
- 4. Based on the **Adjusted Pearson Goodness-of-Fit Test** table, p-values of all group of error term are less than 0.05, indicating they are not followed the assumed distribution well. It suggests the assumed distribution is not good for the data which violate the assumption of the GARCH model.

Estimated GARCH model objects:

```
coef(garchfit_1) # coefficients
```

```
##          mu       omega      alpha1       beta1
## 0.001520781 0.000011263 0.158631065 0.818201410
```

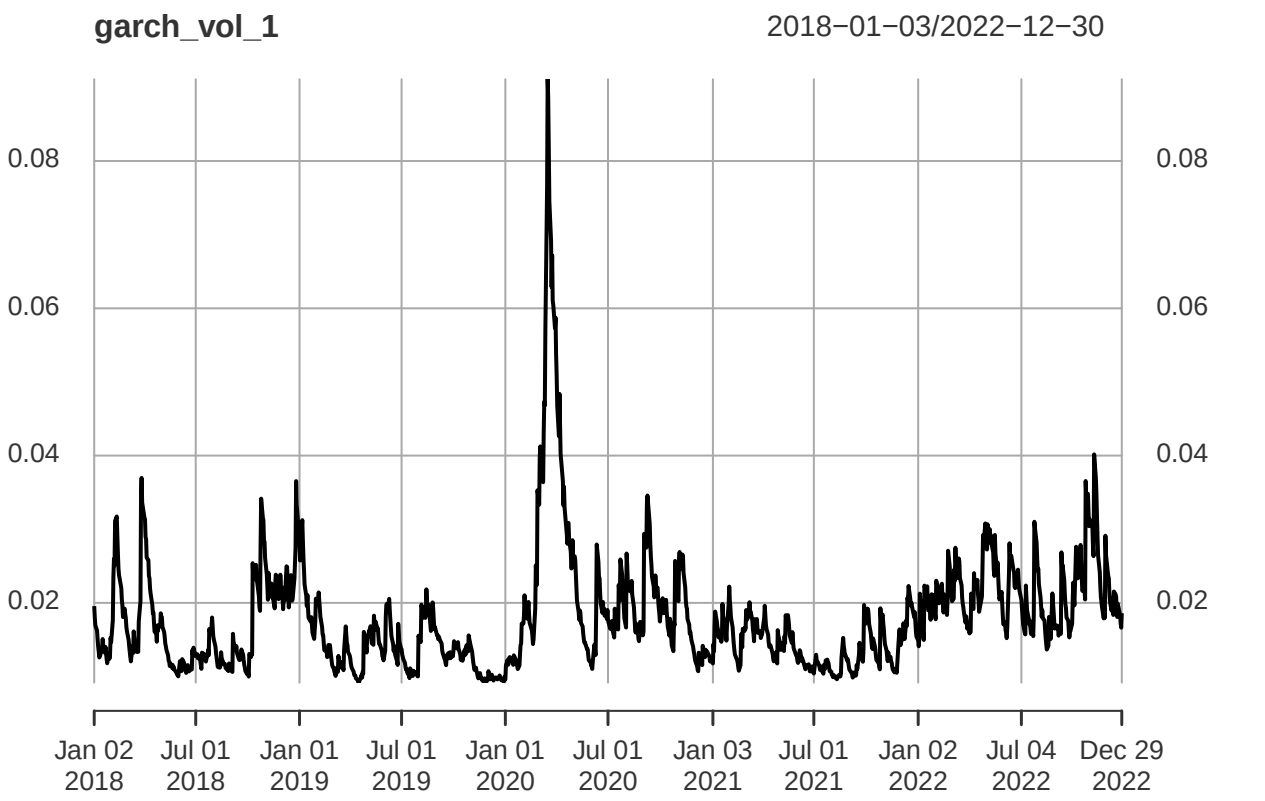
```
uncvariance(garchfit_1) # unconditional variance
```

```
## [1] 0.0004861547
```

This is same as $\frac{\omega}{1-\alpha-\beta}$ which is used to estimated the long-term volatility of the financial returns modeled by the GARCH$(1,1)$.

```
garch_mean_1 <- fitted(garchfit_1) # predicted mean
garch_vol_1 <- sigma(garchfit_1) # predicted volatilities
```

```
plot(garch_vol_1)
```



This plot shows the volatility reverting around 2% in long-run.

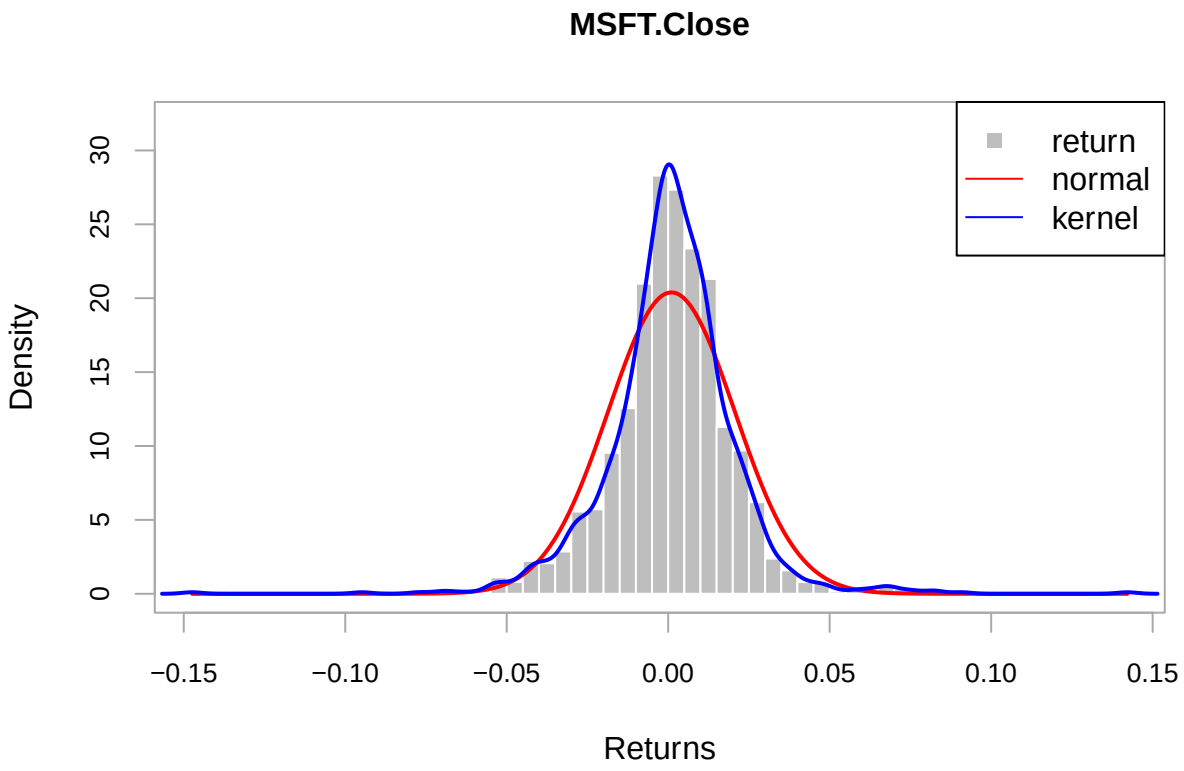
```
# The volatility of next five days
sigma(garchforecast_1)
```

```
##     2022-12-30
## T+1 0.01734453
## T+2 0.01746786
## T+3 0.01758750
## T+4 0.01770358
## T+5 0.01781625
```

However, it is not realistic to assume a normal distribution when analyzing stock returns using a GARCH model. Because it is inconsistent with the phenomenon that stock markets take the stairs up, but the elevator down. This behavior leads to crashes and non-normality in the financial returns.

```
# Testing the normality assumption
chart.Histogram(returns,
                methods = c("add.normal", "add.density"),
                colorset = c("gray", "blue", "red"))
legend("toprigh", legend = c("return", "normal", "kernel"),
       pch = c(15, NA, NA),
       lty = c(NA, 1, 1),
```

```
       col = c("gray", "red", "blue"))
```



This plot shows actual return distribution is much more peaked around zero compared to the normal, indicating the normal distribution is not appropriate to be assumed for the returns.

Therefore, a realistic distribution thus needs to accommodate the potential presence of:

- fat tails: higher probability to observe large (+/-) returns than under the normal distribution
- skewness: asymmetry of the return distribution.

So, change the argument distribution from **"norm"** to **"sstd"** as it has two extra parameters:

- degree of freedom parameter v (in rugarch: shape): the lower is v the fatter the tails.
- Skewness parameter ξ (in rugarch: skew):
  - when $\xi = 1$: symmetry;
  - when $\xi < 1$: negatively skewed;
  - when $\xi > 1$: positively skewed.
- Special cases:
  - when $v = \infty, \xi = 1$: normal distribution
  - when $v = 5, \xi = 1$: student t distribution

GARCH(1, 1) with Skewed Student's t-distribution

$$
\begin{aligned}
R_t &= \mu + e_t \\
e_t &\sim t_v(\xi, \sigma^2) \\
\sigma_t^2 &= \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2
\end{aligned}
$$

```
garchspec_2 <- ugarchspec(mean.model = list(armaOrder = c(0, 0)),
                          variance.model = list(model = "sGARCH"),
                          distribution.model = "sstd")
```

```r
garchfit_2 <- ugarchfit(data = returns, spec = garchspec_2)

garchforecast_2 <- ugarchforecast(fitORspec = garchfit_2, n.ahead = 5)
```

```r
garchfit_2
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001530    0.000409   3.7407 0.000184
## omega   0.000011    0.000009   1.1614 0.245481
## alpha1  0.160837    0.051597   3.1172 0.001826
## beta1   0.818268    0.030138  27.1510 0.000000
## skew    0.927195    0.041936  22.1100 0.000000
## shape   7.006111    2.298434   3.0482 0.002302
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001530    0.000527  2.90199 0.003708
## omega   0.000011    0.000035  0.31018 0.756422
## alpha1  0.160837    0.183968  0.87427 0.381972
## beta1   0.818268    0.074303 11.01260 0.000000
## skew    0.927195    0.085440 10.85205 0.000000
## shape   7.006111    7.355461  0.95251 0.340841
##
## LogLikelihood : 3380.74
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3652
## Bayes        -5.3407
## Shibata      -5.3653
## Hannan-Quinn -5.3560
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      6.044 0.01396
## Lag[2*(p+q)+(p+q)-1][2]     6.611 0.01512
## Lag[4*(p+q)+(p+q)-1][5]     7.212 0.04606
## d.o.f=0
## H0 : No serial correlation
```

```
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.02168  0.8829
## Lag[2*(p+q)+(p+q)-1][5]   0.49718  0.9581
## Lag[4*(p+q)+(p+q)-1][9]   2.05685  0.8981
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1347 0.500 2.000  0.7136
## ARCH Lag[5]    1.0177 1.440 1.667  0.7277
## ARCH Lag[7]    1.6129 2.315 1.543  0.7985
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  10.2632
## Individual Statistics:
## mu     0.15603
## omega  1.10269
## alpha1 0.12843
## beta1  0.38971
## skew   0.22975
## shape  0.08649
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias          0.05111 0.95925
## Negative Sign Bias 1.74112 0.08191   *
## Positive Sign Bias 0.75753 0.44888
## Joint Effect       6.26182 0.09954   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     22.22       0.2733
## 2    30     27.52       0.5438
## 3    40     39.04       0.4679
## 4    50     53.92       0.2917
##
##
## Elapsed time : 0.370156
```

Summary:

- 1. Based on the **Optimal Parameters** table, omega has p-values higher than 0.05, indicating the intercept of the model is close to 0 and not statistically significant to the model.

- 2. **LogLikelihood: 3380.74** and **AIC: -5.3652**
- 3. Based on the **Weighted Ljung-Box Test on Standardized Residuals** table, p-values of tested lags are less than 0.05, indicating there are significant autocorrelation. It suggests the model is not a good fit.
- 4. Based on the **Adjusted Pearson Goodness-of-Fit Test** table, p-values of groups of error terms are slightly better than the GARCH(1, 1) model with constant mean. However, most of p-values are still less than 0.05. It suggests the skewed student t distribution assumption performed better than the normal distribution assumption.

Nevertheless, the sGARCH model utilizes the squared prediction error to predict return variance through a single equation, without differentiating between positive and negative prediction errors. However, in reality, it is crucial to acknowledge that variance tends to increase more significantly following a substantial negative unexpected return compared to a large positive unexpected return.

Negative returns induce higher leverage:

- $R_t < 0$
- market value drop
- leverage increase = debt / market value
- more volatility, more risky

Separate equations for the variance following negative and positive unexpected return $e_t = R_t - \mu_t$:

- when $e_{t-1} > 0$: $\sigma_t^2 = \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2$ usual GARCH model
- when $e_{t-1} < 0$: $\sigma_t^2 = \omega + (\alpha + \gamma) e_{t-1}^2 + \beta \sigma_{t-1}^2$ Employing a larger multiplier on the squared prediction error because it is to reflect the impact of negative surprises. When there's bad news causing negative surprises, the model predicts that the variance should increase more compared to when there's good news and positive surprises. This highlights that negative information about returns has a stronger impact on variance than positive information. The relation is usually explained by the increased leverage ratio that arises from a drop in the share price for a firm.

To define the GJR GARCH model, put these two models together.

GJR-GARCH(1, 1) model with Skewed Student's t-distribution

$$
\begin{aligned}
R_t &= \mu + e_t \\
e_t &\sim t_v(\xi, \sigma^2) \\
\sigma_t^2 &= \begin{cases} \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2, & \text{if } e_{t-1} > 0 \\ \omega + (\alpha + \gamma) e_{t-1}^2 + \beta \sigma_{t-1}^2, & \text{if } e_{t-1} < 0 \end{cases}
\end{aligned}
$$

```
garchspec_3 <- ugarchspec(mean.model = list(armaOrder = c(0, 0)),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_3 <- ugarchfit(data = returns, spec = garchspec_3)

garchforecast_3 <- ugarchforecast(fitORspec = garchfit_3, n.ahead = 5)
```

```
garchfit_3
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
```

```
## ------------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000964    0.000374   2.5786 0.009919
## omega   0.000012    0.000001  12.7889 0.000000
## alpha1  0.030496    0.009524   3.2019 0.001365
## beta1   0.833737    0.018262  45.6554 0.000000
## gamma1  0.213739    0.041191   5.1889 0.000000
## skew    0.897761    0.035637  25.1919 0.000000
## shape   7.234087    1.343356   5.3851 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000964    0.000340   2.8388 0.004529
## omega   0.000012    0.000001  10.5487 0.000000
## alpha1  0.030496    0.013877   2.1976 0.027975
## beta1   0.833737    0.018889  44.1391 0.000000
## gamma1  0.213739    0.044460   4.8075 0.000002
## skew    0.897761    0.033692  26.6458 0.000000
## shape   7.234087    1.379373   5.2445 0.000000
##
## LogLikelihood : 3393.316
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3836
## Bayes        -5.3551
## Shibata      -5.3837
## Hannan-Quinn -5.3729
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      5.809 0.01594
## Lag[2*(p+q)+(p+q)-1][2]     6.231 0.01909
## Lag[4*(p+q)+(p+q)-1][5]     6.689 0.06167
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2351  0.6277
## Lag[2*(p+q)+(p+q)-1][5]    0.5601  0.9480
## Lag[4*(p+q)+(p+q)-1][9]    2.1549  0.8857
## d.o.f=2
##
## Weighted ARCH LM Tests
```

```
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1335 0.500 2.000  0.7149
## ARCH Lag[5]    0.7280 1.440 1.667  0.8150
## ARCH Lag[7]    1.6359 2.315 1.543  0.7937
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  22.9161
## Individual Statistics:
## mu     0.0549
## omega  4.3957
## alpha1 0.1175
## beta1  0.3464
## gamma1 0.1377
## skew   0.2696
## shape  0.0848
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value   prob sig
## Sign Bias          0.004968 0.9960
## Negative Sign Bias 0.739710 0.4596
## Positive Sign Bias 0.352678 0.7244
## Joint Effect       0.746479 0.8622
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     16.94       0.5936
## 2    30     23.65       0.7458
## 3    40     36.82       0.5699
## 4    50     47.41       0.5379
##
##
## Elapsed time : 1.366739
```

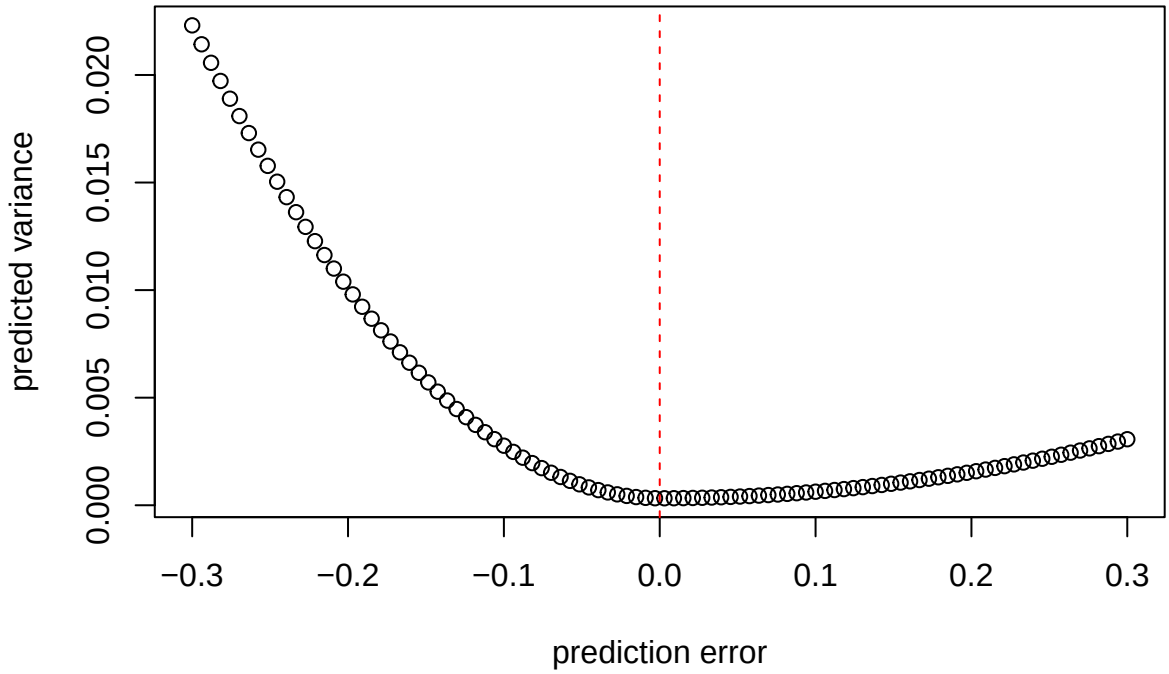
Summary:

- 1. Based on the **Optimal Parameters** table, all parameters have p-values less than 0.05, indicating they are off from 0 and statistically significant to the model.
- 2. **LogLikelihood: 3393.316** and **AIC: -5.3836**
- 3. Based on the **Weighted Ljung-Box Test on Standardized Residuals** table, p-values of tested lags 1 and 6 are less than 0.05, indicating there are significant autocorrelation. It suggests the model is not quite a good fit.
- 4. Based on the **Adjusted Pearson Goodness-of-Fit Test** table, p-values of all groups of error terms are greater than 0.05, indicating the error terms followed the assumed distribution well.

In overall, it performed better than GARCH(1, 1) with Normal distribution and standard GARCH(1, 1) with

Skewed Student's t-distribution.

```
out <- newsimpact(garchfit_3)
plot(out$zx, out$zy,
     xlab = "prediction error",
     ylab = "predicted variance")
abline(v = 0,
       col = "red",
       lty = 2)
```



The News Impact Curve (NIC) depicts the extent of asymmetry in volatility resulting from positive and negative shocks of equal magnitude. According to this plot, the impact of negative shocks on price volatility was significantly more noticeable compared to positive shocks of equivalent magnitude.

GARCH-M with GJR-GARCH(1, 1) model with Skewed Student's t-distribution

It uses the financial theory of a risk-reward trade-off to build a conditional mean model. It incorporates the idea that there is a relationship between risk and expected return.

$$\mu_t = \mu + \lambda \sigma_t^2$$

The risk-reward trade-off parameter λ quantifies the increase in expected return per unit of variance risk. A positive λ implies a positive relationship between risk σ_t^2 and expected return μ_t.

$$\begin{aligned} R_t &= \mu + e_t \\ e_t &\sim t_v(\xi, \sigma^2) \\ \sigma_t^2 &= \begin{cases} \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2, & \text{if } e_{t-1} > 0 \\ \omega + (\alpha + \gamma) e_{t-1}^2 + \beta \sigma_{t-1}^2, & \text{if } e_{t-1} < 0 \end{cases} \end{aligned}$$

```
garchspec_4 <- ugarchspec(mean.model = list(armaOrder = c(0, 0),
                                            archm = TRUE,
```

```
                                        archpow = 2),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_4 <- ugarchfit(data = returns, spec = garchspec_4)

garchforecast_4 <- ugarchforecast(fitORspec = garchfit_4, n.ahead = 5)
```

```
garchfit_4
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000616    0.000411   1.5006 0.133447
## archm   1.569925    1.410737   1.1128 0.265777
## omega   0.000014    0.000001  13.2450 0.000000
## alpha1  0.028328    0.012923   2.1920 0.028378
## beta1   0.827890    0.019095  43.3565 0.000000
## gamma1  0.215740    0.041808   5.1602 0.000000
## skew    0.897189    0.035891  24.9974 0.000000
## shape   7.260238    1.367428   5.3094 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000616    0.000386   1.5966 0.110353
## archm   1.569925    1.133288   1.3853 0.165966
## omega   0.000014    0.000001  10.9860 0.000000
## alpha1  0.028328    0.014595   1.9410 0.052263
## beta1   0.827890    0.020145  41.0967 0.000000
## gamma1  0.215740    0.043112   5.0042 0.000001
## skew    0.897189    0.033447  26.8245 0.000000
## shape   7.260238    1.363361   5.3252 0.000000
##
## LogLikelihood : 3393.631
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3826
## Bayes        -5.3499
## Shibata      -5.3826
## Hannan-Quinn -5.3703
##
```

```
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      5.035 0.02484
## Lag[2*(p+q)+(p+q)-1][2]     5.358 0.03263
## Lag[4*(p+q)+(p+q)-1][5]     5.679 0.10694
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2355  0.6275
## Lag[2*(p+q)+(p+q)-1][5]    0.5106  0.9560
## Lag[4*(p+q)+(p+q)-1][9]    2.0099  0.9038
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     0.109 0.500 2.000  0.7413
## ARCH Lag[5]     0.627 1.440 1.667  0.8456
## ARCH Lag[7]     1.477 2.315 1.543  0.8260
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  21.9835
## Individual Statistics:
## mu     0.06559
## archm  0.07045
## omega  3.02584
## alpha1 0.11223
## beta1  0.33351
## gamma1 0.11545
## skew   0.28288
## shape  0.08087
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.2348 0.8144
## Negative Sign Bias  0.4612 0.6448
## Positive Sign Bias  0.6280 0.5301
## Joint Effect        0.6467 0.8857
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     20.06       0.3910
```

```
## 2    30     32.00       0.3199
## 3    40     36.75       0.5728
## 4    50     50.27       0.4230
##
##
## Elapsed time : 1.663018
```

Summary:

- 1. Based on the **Optimal Parameters** table, both mu and archm parameters have p-values higher than 0.05, indicating they are close to 0 and not statistically significant to the model.
- 2. **LogLikelihood: 3393.631** and **AIC: -5.3826**
- 3. Based on the **Weighted Ljung-Box Test on Standardized Residuals** table, all p-values of tested lags are greater than 0.05, indicating there are no significant autocorrelation. It suggests the model is a good fit.
- 4. Based on the **Adjusted Pearson Goodness-of-Fit Test** table, p-values of all groups of error terms are greater than 0.05, indicating the error terms followed the assumed distribution well.

Zoom into 2020 Covid year for understanding the risk-reward trade-off

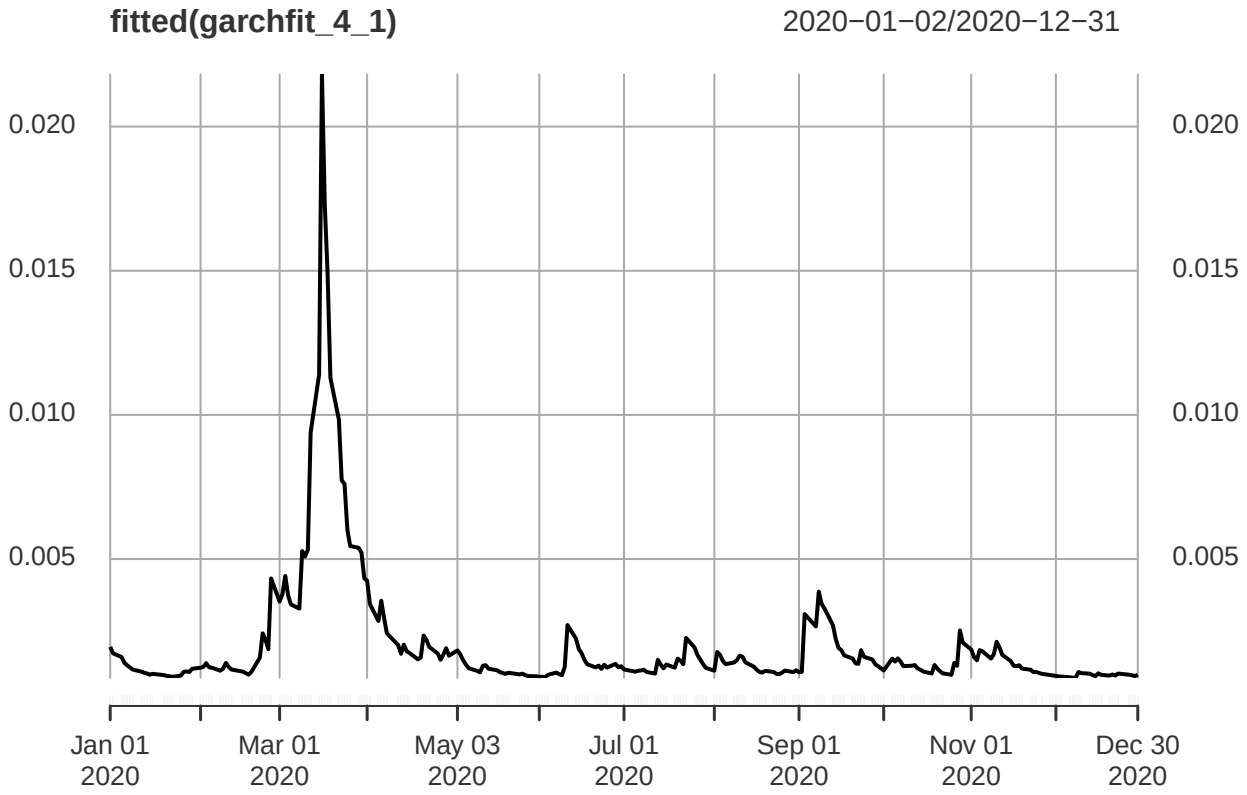
```
garchfit_4_1 <- ugarchfit(data = returns["2020"], spec = garchspec_4)

round(coef(garchfit_4_1)[1:2], 4)
```

```
##     mu  archm
## 0.0006 1.7697
```

$$\mu_t = \mu + \lambda\sigma_t^2 = 0.0006 + 1.7697\sigma_t^2$$

For each additional unit of the variance risk, the expected return increases by 1.7697 units.

```
plot(fitted(garchfit_4_1))
```



Spike around in 2020 April. Investors require a higher return in compensation for the high risk.

```
chartSeries(MSFT["2020-4"])
```

ARMA(#, #)-GJR-GARCH(1, 1) model with Skewed Student's t-distribution

Besides the financial theory of a risk-reward trade-off, the statistical theory of building a conditional mean model considers the autocorrelation structure in returns. It relates to the correlation between today's and tomorrow's returns.

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

It predicts the next return using the deviation of the return from its long term mean value μ.

$$\mu_t = \mu + \rho(R_{t-1} - \mu)$$

The autoregressive coefficient ρ:

- when $\rho > 0$: A higher/lower than average return is followed by a higher/lower than average return. It may suggest markets underreact to news. (+, +) / (-, -)
- when $|\rho| < 1$: mean reversion: the deviation of R_t from μ are transitory.
- when $\rho < 0$: A higher/lower than average return is followed by a lower/higher than average return. It may suggests markets overreact to news. (+, -) / (-, +)

```
garchspec_5 <- ugarchspec(mean.model = list(armaOrder = c(1, 0)),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_5 <- ugarchfit(data = returns, spec = garchspec_5)

garchforecast_5 <- ugarchforecast(fitORspec = garchfit_5, n.ahead = 5)
```

```
garchfit_5
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001078    0.000347   3.1023 0.001920
## ar1    -0.094675    0.029292  -3.2321 0.001229
## omega   0.000012    0.000001  12.0074 0.000000
## alpha1  0.032149    0.008875   3.6222 0.000292
## beta1   0.842121    0.017499  48.1253 0.000000
## gamma1  0.190542    0.037657   5.0600 0.000000
## skew    0.886521    0.035815  24.7530 0.000000
## shape   7.348889    1.404716   5.2316 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001078    0.000330   3.2617 0.001107
## ar1    -0.094675    0.028088  -3.3707 0.000750
## omega   0.000012    0.000001  10.0726 0.000000
## alpha1  0.032149    0.013616   2.3611 0.018223
## beta1   0.842121    0.017163  49.0652 0.000000
## gamma1  0.190542    0.039805   4.7869 0.000002
## skew    0.886521    0.034131  25.9738 0.000000
## shape   7.348889    1.364058   5.3875 0.000000
##
## LogLikelihood : 3398.477
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3903
## Bayes        -5.3576
## Shibata      -5.3903
## Hannan-Quinn -5.3780
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1673  0.6825
## Lag[2*(p+q)+(p+q)-1][2]    0.8240  0.8392
## Lag[4*(p+q)+(p+q)-1][5]    1.5511  0.8340
## d.o.f=1
```

```
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3093  0.5781
## Lag[2*(p+q)+(p+q)-1][5]    0.5197  0.9546
## Lag[4*(p+q)+(p+q)-1][9]    1.9719  0.9083
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.08089 0.500 2.000  0.7761
## ARCH Lag[5]   0.45164 1.440 1.667  0.8978
## ARCH Lag[7]   1.24250 2.315 1.543  0.8712
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  22.244
## Individual Statistics:
## mu     0.05864
## ar1    0.57624
## omega  4.04928
## alpha1 0.13578
## beta1  0.35145
## gamma1 0.16526
## skew   0.27862
## shape  0.10233
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value   prob sig
## Sign Bias          0.003632 0.9971
## Negative Sign Bias 0.793455 0.4277
## Positive Sign Bias 0.082101 0.9346
## Joint Effect       0.863804 0.8342
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     16.91       0.5958
## 2    30     23.84       0.7366
## 3    40     34.08       0.6935
## 4    50     45.10       0.6320
##
##
## Elapsed time : 1.35473
```

```
round(coef(garchfit_5)[1:2], 4)
```

```
##      mu     ar1
##  0.0011 -0.0947
```

The negative AR(1) coefficient may suggest a tendency for markets to overreact to extreme returns by exhibiting a mean-reverting behavior.

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

It predicts the next return using the deviation of the return from its conditional mean value μ.

$$\mu_t = \mu + \theta(R_{t-1} - \mu_{t-1})$$

The moving average coefficient θ:

- when $\theta > 0$: A positive innovation in the past period is associated with a higher-than-average return in the current period. It may suggest markets underreact to news.
- when $|\theta| < 1$: mean reversion: the deviation of R_{t-1} from μ_{t-1} are transitory.
- when $\theta < 0$: a negative innovation in the past period is associated with a higher-than-average return in the current period. It may suggests markets overreact to news.

```
garchspec_6 <- ugarchspec(mean.model = list(armaOrder = c(0, 1)),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_6 <- ugarchfit(data = returns, spec = garchspec_6)

garchforecast_6 <- ugarchforecast(fitORspec = garchfit_6, n.ahead = 5)

garchfit_6
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(0,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001097    0.000343   3.2001 0.001374
## ma1    -0.103637    0.030568  -3.3903 0.000698
## omega   0.000012    0.000001  11.9278 0.000000
## alpha1  0.032880    0.008858   3.7118 0.000206
## beta1   0.842871    0.017423  48.3780 0.000000
## gamma1  0.186889    0.037346   5.0043 0.000001
## skew    0.884361    0.035877  24.6501 0.000000
## shape   7.350139    1.407118   5.2235 0.000000
##
```

```
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001097    0.000330   3.3270 0.000878
## ma1    -0.103637    0.030222  -3.4292 0.000605
## omega   0.000012    0.000001   9.9700 0.000000
## alpha1  0.032880    0.013710   2.3982 0.016475
## beta1   0.842871    0.016967  49.6763 0.000000
## gamma1  0.186889    0.039321   4.7529 0.000002
## skew    0.884361    0.034394  25.7126 0.000000
## shape   7.350139    1.358867   5.4090 0.000000
##
## LogLikelihood : 3398.958
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3910
## Bayes        -5.3584
## Shibata      -5.3911
## Hannan-Quinn -5.3788
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4627  0.4964
## Lag[2*(p+q)+(p+q)-1][2]    0.8185  0.8420
## Lag[4*(p+q)+(p+q)-1][5]    1.3888  0.8729
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3316  0.5647
## Lag[2*(p+q)+(p+q)-1][5]    0.5320  0.9526
## Lag[4*(p+q)+(p+q)-1][9]    1.9772  0.9077
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.07482 0.500 2.000  0.7844
## ARCH Lag[5]   0.43477 1.440 1.667  0.9027
## ARCH Lag[7]   1.20112 2.315 1.543  0.8788
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  22.2026
## Individual Statistics:
## mu     0.0604
## ma1    0.5758
## omega  4.0362
## alpha1 0.1376
## beta1  0.3491
```

```
## gamma1 0.1689
## skew   0.2752
## shape  0.1069
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value   prob sig
## Sign Bias          0.06382 0.9491
## Negative Sign Bias 0.83574 0.4035
## Positive Sign Bias 0.03597 0.9713
## Joint Effect       0.90374 0.8245
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     14.21       0.7713
## 2    30     26.52       0.5978
## 3    40     29.12       0.8754
## 4    50     35.32       0.9288
##
##
## Elapsed time : 1.091314
```

```
round(coef(garchfit_6)[1:2], 4)
```

```
##      mu     ma1
##  0.0011 -0.1036
```

The negative MA(1) coefficient suggests that the model incorporates information from the past innovation term in predicting the current return. The negative innovation in the previous period suggests that the subsequent return is expected to be lower than the baseline expected return.

ARMA(1, 1)-GJR-GARCH(1, 1)

The combination of AR(1) and MA(1)

$$\mu_t = \mu + \rho(R_{t-1} - \mu) + \theta(R_{t-1} - \mu_{t-1})$$

```
garchspec_7 <- ugarchspec(mean.model = list(armaOrder = c(1, 1)),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_7 <- ugarchfit(data = returns, spec = garchspec_7)

garchforecast_7 <- ugarchforecast(fitORspec = garchfit_7, n.ahead = 5)

garchfit_7
```

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

```
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ---------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001269    0.000300   4.2370 0.000023
## ar1     0.682521    0.202060   3.3778 0.000731
## ma1    -0.769473    0.176686  -4.3550 0.000013
## omega   0.000012    0.000001  11.1793 0.000000
## alpha1  0.043174    0.009967   4.3318 0.000015
## beta1   0.840241    0.017773  47.2759 0.000000
## gamma1  0.166420    0.036750   4.5285 0.000006
## skew    0.882738    0.035937  24.5632 0.000000
## shape   7.141347    1.328711   5.3746 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001269    0.000327   3.8786 0.000105
## ar1     0.682521    0.341488   1.9987 0.045644
## ma1    -0.769473    0.300229  -2.5630 0.010379
## omega   0.000012    0.000001   9.0870 0.000000
## alpha1  0.043174    0.016752   2.5772 0.009959
## beta1   0.840241    0.017368  48.3777 0.000000
## gamma1  0.166420    0.040108   4.1493 0.000033
## skew    0.882738    0.034811  25.3582 0.000000
## shape   7.141347    1.346885   5.3021 0.000000
##
## LogLikelihood : 3401.839
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3940
## Bayes        -5.3573
## Shibata      -5.3941
## Hannan-Quinn -5.3802
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.03997  0.8415
## Lag[2*(p+q)+(p+q)-1][5]   1.45459  0.9986
## Lag[4*(p+q)+(p+q)-1][9]   2.57905  0.9440
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
```

```
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1773  0.6737
## Lag[2*(p+q)+(p+q)-1][5]    0.4550  0.9644
## Lag[4*(p+q)+(p+q)-1][9]    1.9803  0.9073
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.01542 0.500 2.000  0.9012
## ARCH Lag[5]   0.47966 1.440 1.667  0.8896
## ARCH Lag[7]   1.21993 2.315 1.543  0.8754
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  21.7752
## Individual Statistics:
## mu     0.1029
## ar1    0.5096
## ma1    0.4615
## omega  4.0061
## alpha1 0.1284
## beta1  0.3263
## gamma1 0.1616
## skew   0.2618
## shape  0.1269
##
## 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          0.27377 0.7843
## Negative Sign Bias 0.55691 0.5777
## Positive Sign Bias 0.07274 0.9420
## Joint Effect       0.82598 0.8432
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     19.39       0.4319
## 2    30     35.62       0.1848
## 3    40     44.26       0.2594
## 4    50     52.97       0.3236
##
##
## Elapsed time : 1.413714
```

```
round(coef(garchfit_7)[1:2], 4)
```

```
##     mu    ar1
```

```
## 0.0013 0.6825
```

The positive MA(1) coefficient suggests that the model incorporates information from the past innovation term in predicting the current return. The positive innovation in the previous period suggests that the subsequent return is expected to be higher than the baseline expected return.

Models Comparison

```
GARCH_models <- data.frame(
  Models = c("n_(1,1)", "s_(1,1)", "gjr_(1,1)", "gjr_(1,1)-M", "AR(1)", "MA(1)", "ARMA(1,1)"),
  Insig_parameters = c("None", "omega", "None", "mu & archm", "None", "None", "None"),
  Log_likelihood = c(3356.619, 3380.74, 3393.316, 3393.631, 3398.477, 3398.958, 3401.839),
  AIC = c(-5.3301, -5.3652, -5.3836, -5.3826, -5.3903, -5.3910, -5.3940),
  BIC = c(-5.3137, -5.3407, -5.3551, -5.3499, -5.3576, -5.3584, -5.3573),
  Insig_Ljung_Box = c(0, 0, 1, 1, 3, 3, 3),
  Insig_Goodness_of_fit = c(0, 1, 4, 4, 4, 4, 4)
  )

GARCH_models
```

```
##        Models Insig_parameters Log_likelihood     AIC     BIC Insig_Ljung_Box
## 1     n_(1,1)             None       3356.619 -5.3301 -5.3137               0
## 2     s_(1,1)            omega       3380.740 -5.3652 -5.3407               0
## 3   gjr_(1,1)             None       3393.316 -5.3836 -5.3551               1
## 4 gjr_(1,1)-M       mu & archm       3393.631 -5.3826 -5.3499               1
## 5       AR(1)             None       3398.477 -5.3903 -5.3576               3
## 6       MA(1)             None       3398.958 -5.3910 -5.3584               3
## 7   ARMA(1,1)             None       3401.839 -5.3940 -5.3573               3
##   Insig_Goodness_of_fit
## 1                     0
## 2                     1
## 3                     4
## 4                     4
## 5                     4
## 6                     4
## 7                     4
```

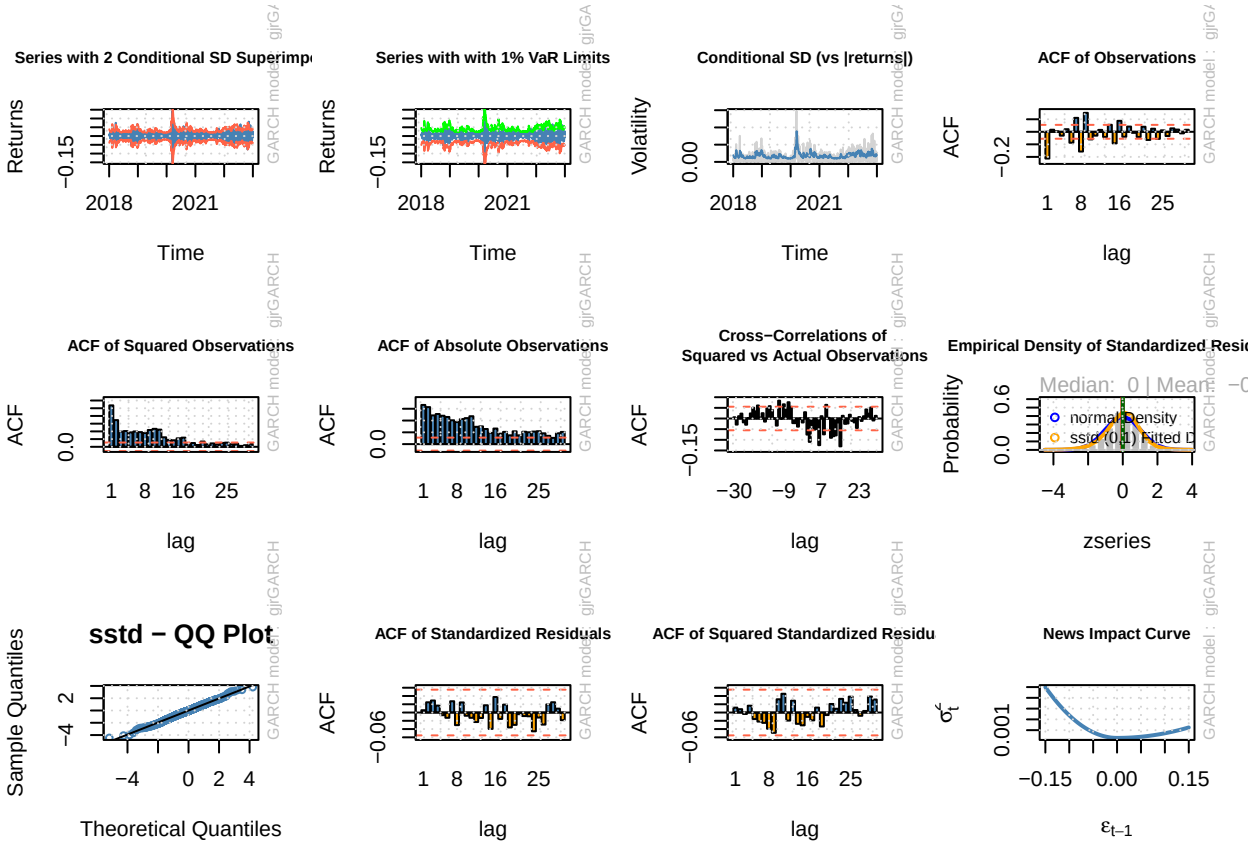
Variables:

- Models: 7 GARCH models in this section.
- Insig_parameters: p-values < 0.05 for each coefficient from the **Optimal Parameters** table. The p-values are associated with t-tests that assess whether the estimated coefficients are significantly different from zero.
- Log_likelihood: the higher, the better model.
- AIC: the lower, the better model.
- BIC: the lower, the better model.
- Insig_Ljung_Box: the number of the insignificant tested lags from the **Weighted Ljung-Box Test on Standardized Residuals** table. The insignificant tested lags show there are no significant autocorrelation. The more, the better model.
- Insig_Goodness_of_fit: the number of the insignificant group of error term from the **Adjusted Pearson Goodness-of-Fit Test** table. The insignificant group of error shows the conditional error term follows the assumed distribution well. The more, the better assumption of the conditional distribution.

Based on the result, ARMA(1, 1)-GJR-GARCH(1, 1) model with Skewed Student's t-distribution outperforms other models.

```
garchspec_7 <- ugarchspec(mean.model = list(armaOrder = c(1, 1)),
                          variance.model = list(model = "gjrGARCH"),
                          distribution.model = "sstd")

garchfit_7 <- ugarchfit(data = returns, spec = garchspec_7)

garchforecast_7 <- ugarchforecast(fitORspec = garchfit_7, n.ahead = 22)

plot(garchfit_7, which = "all")
```

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




```
coef(garchfit_7) # coefficient
```

```
##            mu           ar1           ma1         omega        alpha1
##  1.269337e-03  6.825212e-01 -7.694732e-01  1.181633e-05  4.317438e-02
##         beta1        gamma1          skew         shape
##  8.402414e-01  1.664205e-01  8.827379e-01  7.141347e+00
```

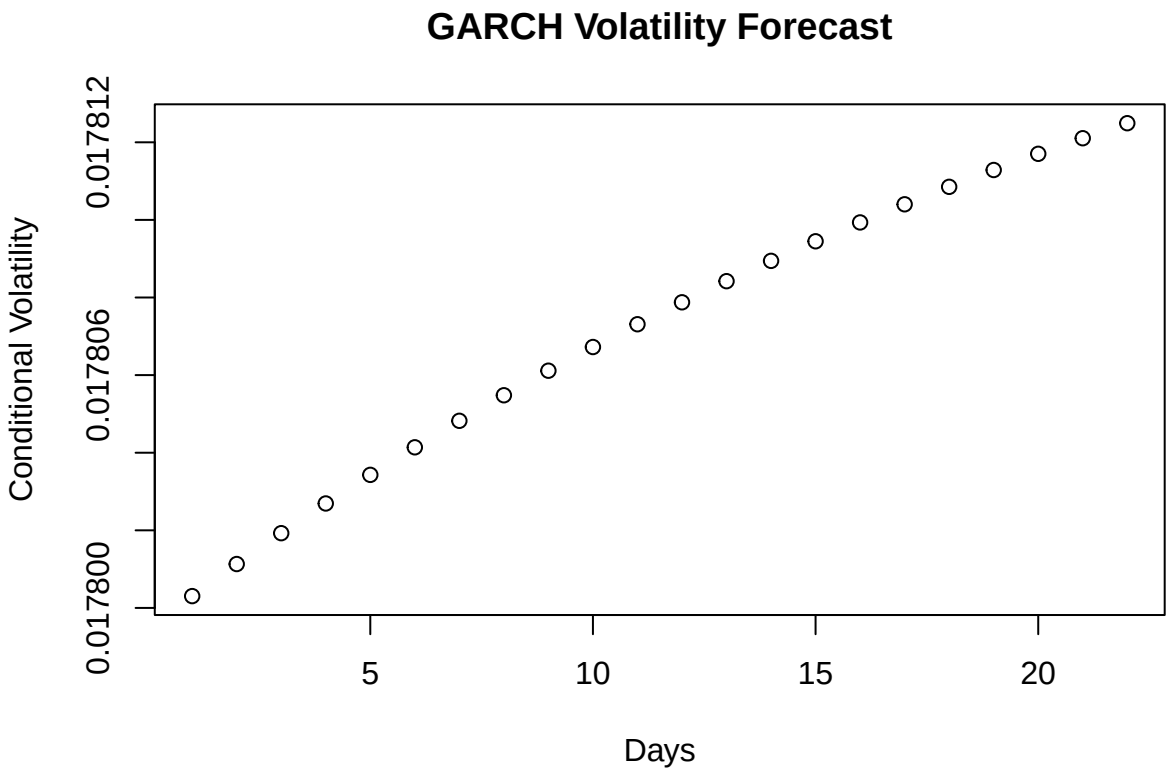
```
uncvariance(garchfit_7) # unconditional variance
```

```
## [1] 0.0003176415
```

```
garch_mean_7 <- fitted(garchfit_7) # predicted mean
garch_vol_7 <- sigma(garchfit_7) # predicted volatilities
```

```
plot(sigma(garchforecast_7),
     ylab = "Conditional Volatility",
```

```
    xlab = "Days",
    main = "GARCH Volatility Forecast")
```



This plot shows the prediction of volatility increase in the next 22 days. The predictive model expects higher variability or risk in the returns of the financial instrument at the beginning of 2023, indicating an expectation of increased price fluctuations or uncertainty in the market.

Backtesting

Backtesting is a process to access the forecasting accuracy of a time-dependent statistical model. It is time series version of cross-validation.

Two concepts related to the backtesting:

- 1. Expand the Window Estimation (use all available returns at the time of prediction)
  - We consider all historical data available up to current when making predictions for a certain point in time. It means the model takes into account information from the entire dataset up to the current moment.
- 2. Moving Window Estimation (Only use a fixed number of the most recent return observations)
  - We set a fixed window size, and only consider the. most recent observations within a fixed window for each prediction point. As new data becomes available, the window moves forward in time.

In the previous section, we applied `sigma()` to the ugarchforecast object for predicting the volatility. However, it has the look ahead bias because the future returns are used to make the volatility estimate. One of the solutions is to apply `ugarchroll()` check the whether the in-sample estimate aligned with rolling estimation.

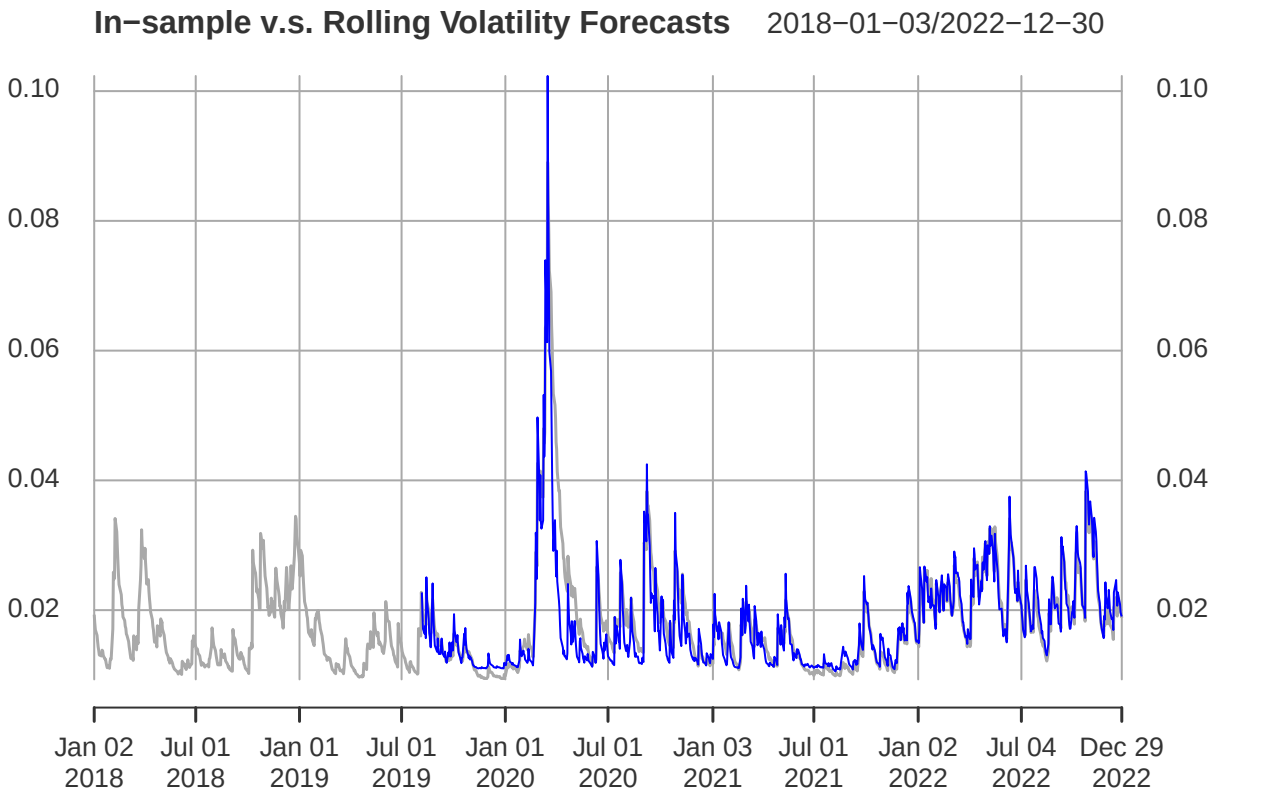
aruguments in `ugarchroll()`:

- GARCH specification used
- data
- n.start: the size of the initial estimation sample

- refit.window: how to change that sample through time, “moving”/“expanding”
- refit.every: how often to re-estimate the model.

In the following practice, we will go through the Moving Window Estimation approach as wee believe the most recent observations weighted more than the past in a real time financial data.

```
garchroll <- ugarchroll(garchspec_7,
                        data = returns,
                        n.start = 400,
                        refit.window = "moving",
                        refit.every = 500)

# Set garchroll_pred to the data frame with rolling predictions
garchroll_pred <- as.data.frame(garchroll)

# Compare In-sample estimate and rolling estimation volatility in one plot
garchroll_vol <- xts(garchroll_pred$Sigma,
                     order.by = as.Date(rownames(garchroll_pred)))

volplot <- plot(garch_vol_7,
                col = "darkgrey",
                lwd = 1.5,
                main = "In-sample v.s. Rolling Volatility Forecasts")
volplot <- addSeries(garchroll_vol,
                     col = "blue",
                     on = 1)
plot(volplot)
```



Notice that the blue line follows the dark grey line quite well which suggests that the rolling volatility forecasts are consistent with the in-sample forecasts.

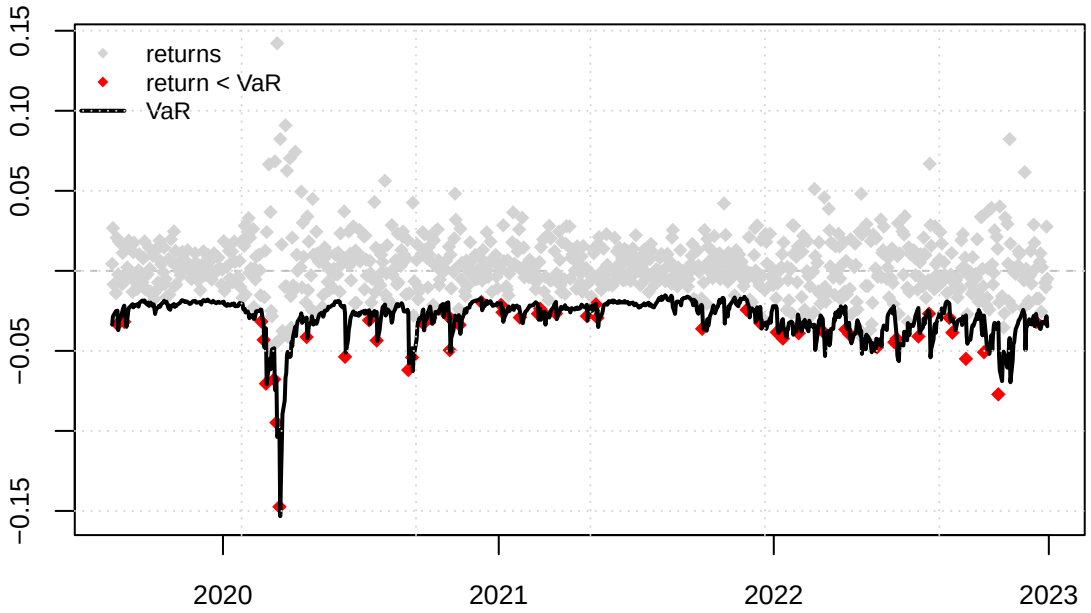
```r
# Compute prediction errors
e  <- garchroll_pred$Realized - garchroll_pred$Mu
d  <- e^2 - garchroll_pred$Sigma^2

# Compute MSE for the garchroll variance prediction
garchMSE <- mean(d^2)
garchMSE # the lower, the better (can be used in comparing other models)
```

```
## [1] 1.215408e-06
```

Value-at-Risk (VaR) is a widely used metric for assessing downside risk. It quantifies the maximum potential loss, at a specified confidence level, within a given time horizon. Specifically, the 5% VaR is commonly interpreted as the threshold below which losses are not expected to exceed 95% of the time. In other words, it represents the worst expected loss in the most adverse 5% of scenarios, providing a measure of the potential downside risk for a financial portfolio or investment.

return $<$ VaR: exceedance

```r
# Compute the predicted quantile
garchvar <- quantile(garchroll, probs = 0.05)
actual <- xts(garchroll_pred$Realized, time(garchvar))
VaRplot(alpha = 0.05, actual = actual, VaR = garchvar)
```




```r
mean(actual < garchvar)
```

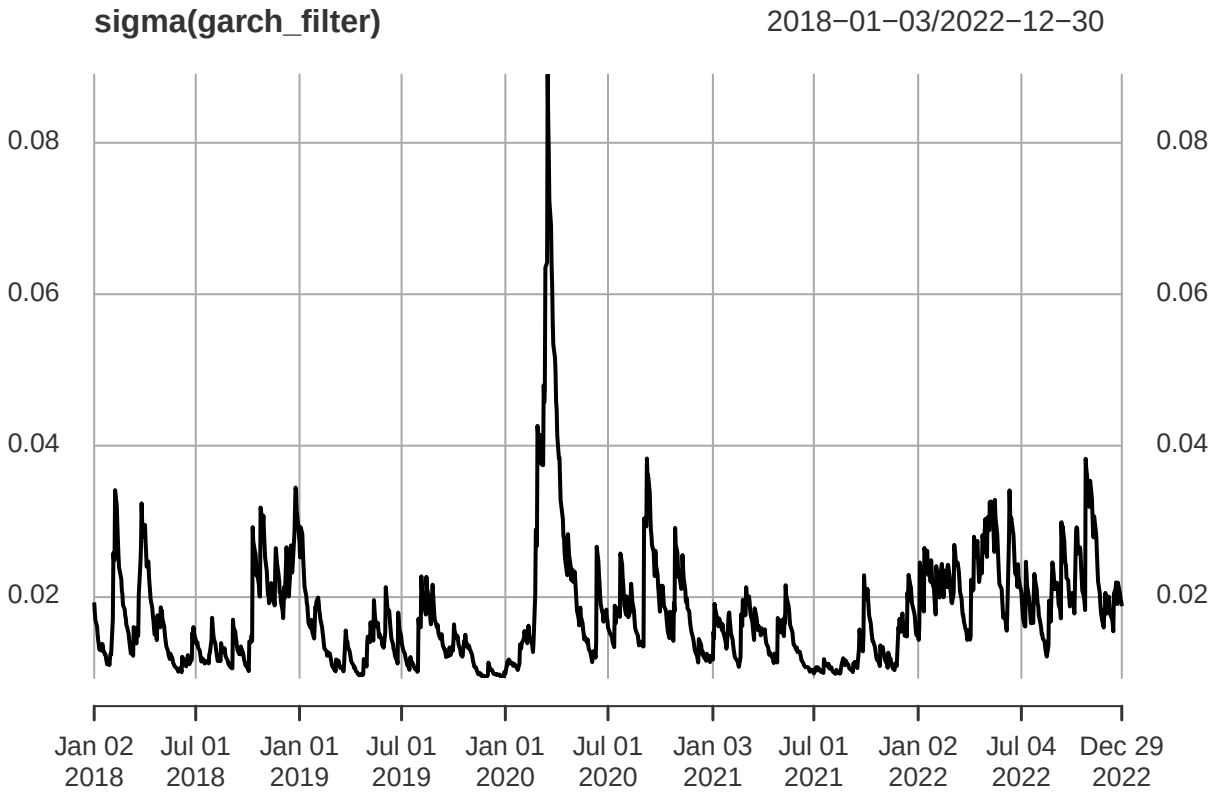
```
## [1] 0.05944056
```

This result shows approximately 5.94% of actual returns falling below the predicted 5% Value-at-Risk (VaR), implies a notable alignment between the model's VaR estimate and the observed data. This result suggests that the model's forecasted downside risk, represented by the 5% VaR, is consistent with the realized returns, reinforcing confidence in the reliability of the VaR estimation for risk management purposes.

Production Setting

In a production setting, models are used to make real-time predictions, manage risk, or inform decision-making in a live, operational context. We can apply `ugarchfilter()` to analyze the current and often dynamic data in the mean and volatility, and to apply `ugarchforeacst()` for `ugarchspec()` instead of `ugarchfit()`.

```r
# re-define the final garchspec
final_garchspec <- garchspec_7
setfixed(final_garchspec) <- as.list(coef(garchfit_7))

garch_filter <- ugarchfilter(data = returns,
                             spec = final_garchspec)

plot(sigma(garch_filter))
```




```r
# Actual forecast
final_forecast <- ugarchforecast(data = returns,
                                 fitORspec = final_garchspec,
                                 n.ahead = 5)

cbind(fitted(final_forecast), sigma(final_forecast))
```

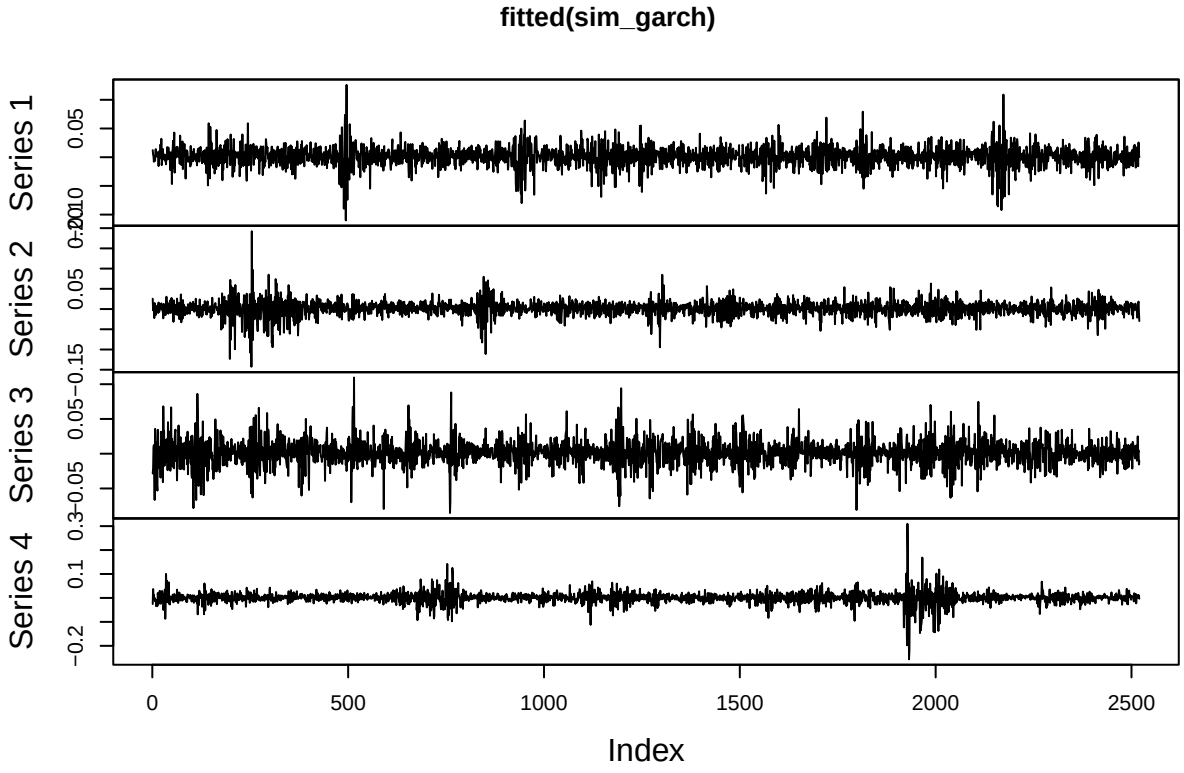
```
##      2022-12-30 2022-12-30
## T+1 0.001798514 0.01780030
## T+2 0.001630512 0.01780113
## T+3 0.001515846 0.01780193
## T+4 0.001437585 0.01780269
## T+5 0.001384170 0.01780343
```

Simulation

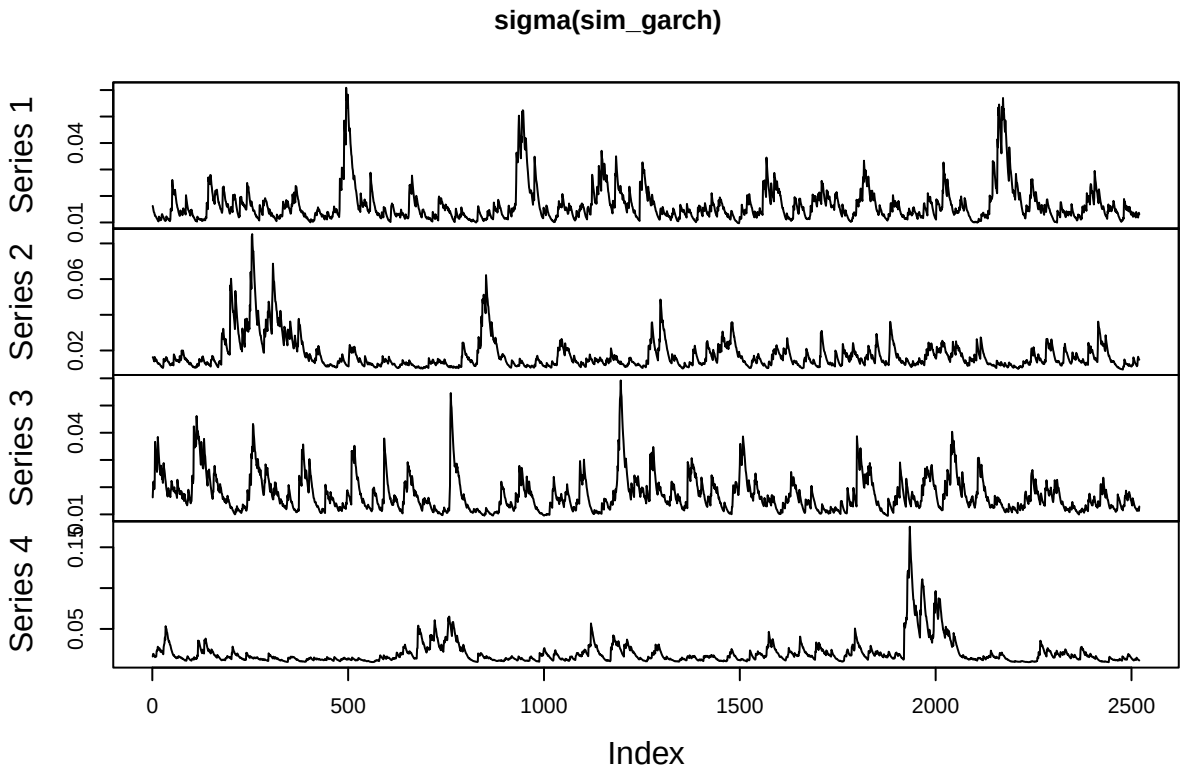
There are several reasons for simulating the model, such as for better understanding and managing risk to make informed decision, testing model assumption, and improving the overall robustness of model. It is useful to assess the randomness in future returns and the impact on prices, since the future price equals: $P_{t+h} = P_t e^{(r_{t+1}+r_{t+2}+\ldots+r_{t+h})}$

Use the log returns $r_t = log(P_t) - log(P_{t-1})$ for simulation as its mathematical properties (additivity of returns and stability of variance) simplifies the aggregation of returns which is the sum of log returns over different time intervals is equal to the log return of the total period and provides more stable variances over time compared to simple returns.

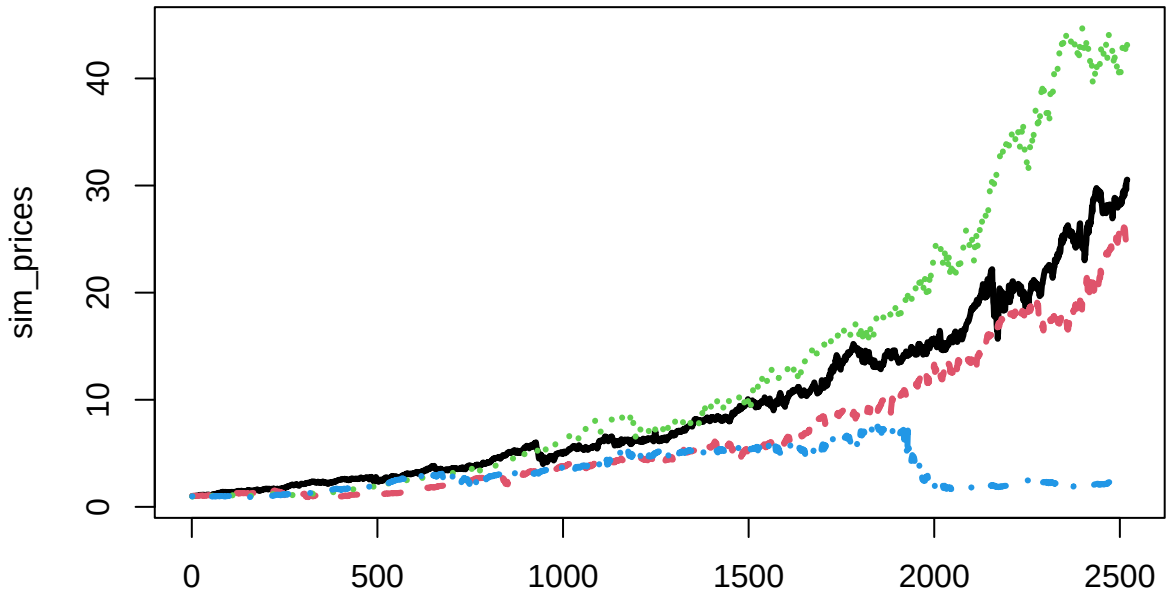
```
log_returns <- diff(log(MSFT$MSFT.Close))[-1]

log_garchspec_7 <- ugarchspec(mean.model = list(armaOrder = c(1, 1)),
                              variance.model = list(model = "gjrGARCH"),
                              distribution.model = "sstd")
log_garchfit_7 <- ugarchfit(data = log_returns, spec = log_garchspec_7)

final_log_garchspec <- log_garchspec_7
setfixed(final_log_garchspec) <- as.list(coef(log_garchfit_7))

sim_garch <- ugarchpath(spec = final_log_garchspec,
                        m.sim = 4, # number of time series of simulated returns
                        n.sim = 10 * 252, # number of observations in each simulated time series
                        rseed = 123)

# Plot the simulated returns
plot.zoo(fitted(sim_garch))
```




```
# Plot the simulated conditional standard deviations
plot.zoo(sigma(sim_garch))
```




```
# Plot the simulated prices over time
sim_prices <- exp(apply(fitted(sim_garch), 2, "cumsum"))
matplot(sim_prices, type = "l", lwd = 3)
```

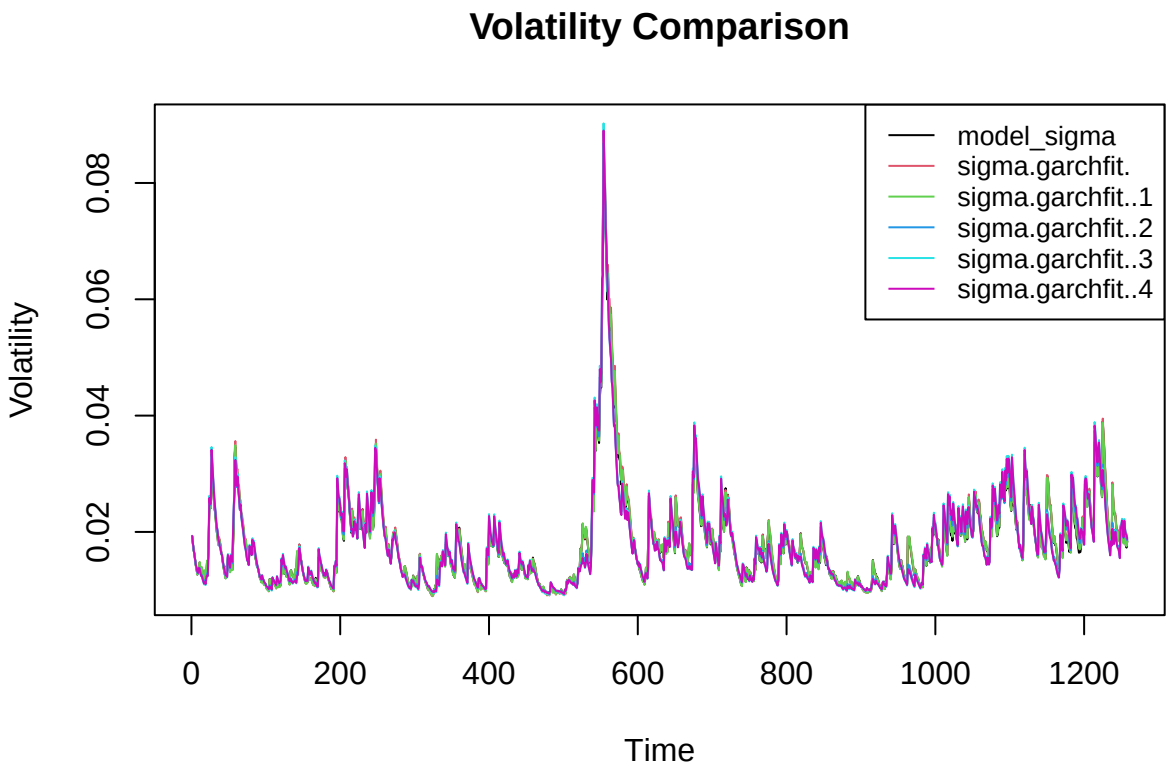


Ensemble Model

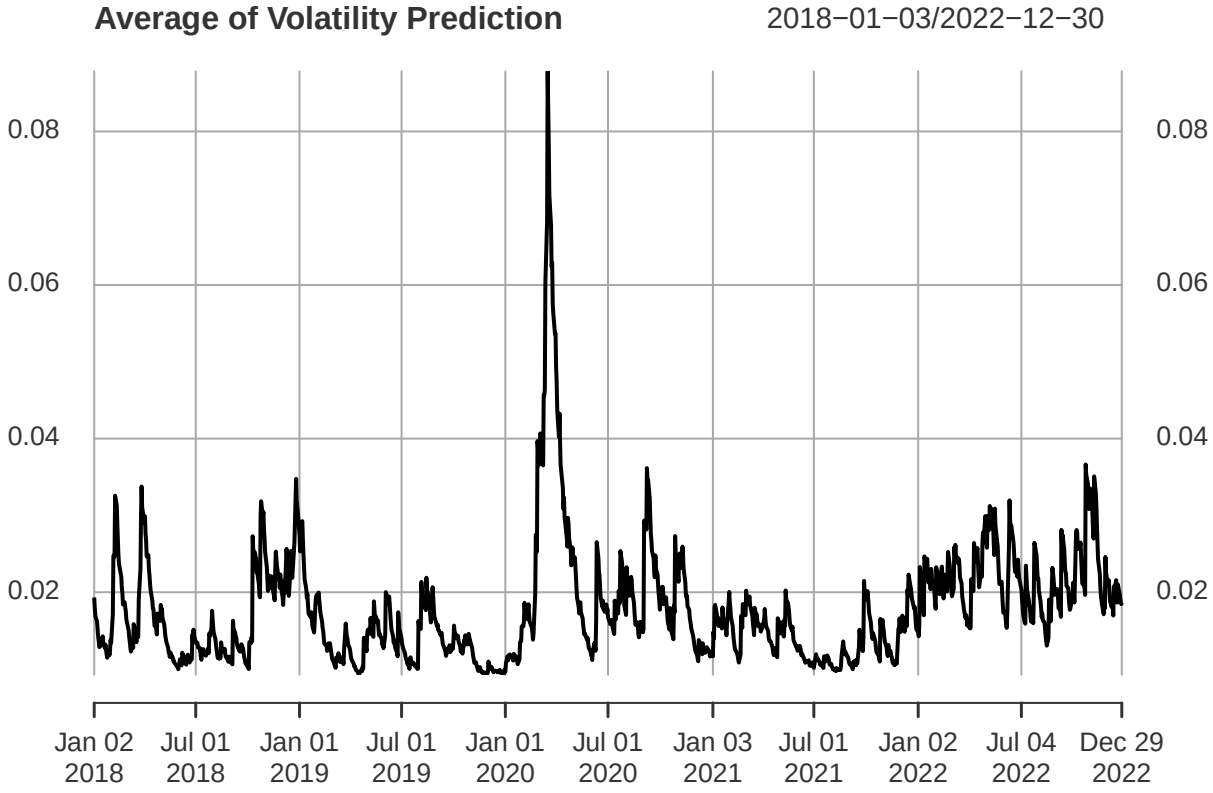
In real-world scenarios, individual models often encounter challenges and complexities arising from dynamic environments and unexpected events, such as the impact of factors like COVID-19 or adverse news. To address these limitations, a robust approach is introduced, involving the utilization of ensemble methods, the amalgamation of predictions from multiple models, and the application of techniques like transfer learning. In the realm of statistical and machine learning models, a robust approach is characterized by its reduced

sensitivity to outliers or noise in the data. Robust statistical methods and models are specifically crafted to deliver accurate results even when confronted with violations of the assumptions inherent in traditional statistical techniques.

```
variance_models <- c("sGARCH", "gjrGARCH")
distribution_models <- c("norm", "std", "sstd")
counter <- 1
for (var.model in variance_models) {
  for (dist.model in distribution_models) {
    garchspec <- ugarchspec(mean.model = list(armaOrder = c(1, 1)),
                            variance.model = list(model = var.model),
                            distribution.model = dist.model)
    garchfit <- ugarchfit(data = returns, spec = garchspec)
    if (counter == 1){
      model_sigma <- sigma(garchfit)
    } else {
      model_sigma <- merge(model_sigma, sigma(garchfit))
      }
    counter <- counter + 1
  }
}

matplot(model_sigma,
        type = "l",
        col = 1:6,
        lty = 1,
        xlab = "Time",
        ylab = "Volatility",
        main = "Volatility Comparison")
legend("topright",
       legend = colnames(model_sigma),
       col = 1:6, lty = 1, cex = 0.8)
```



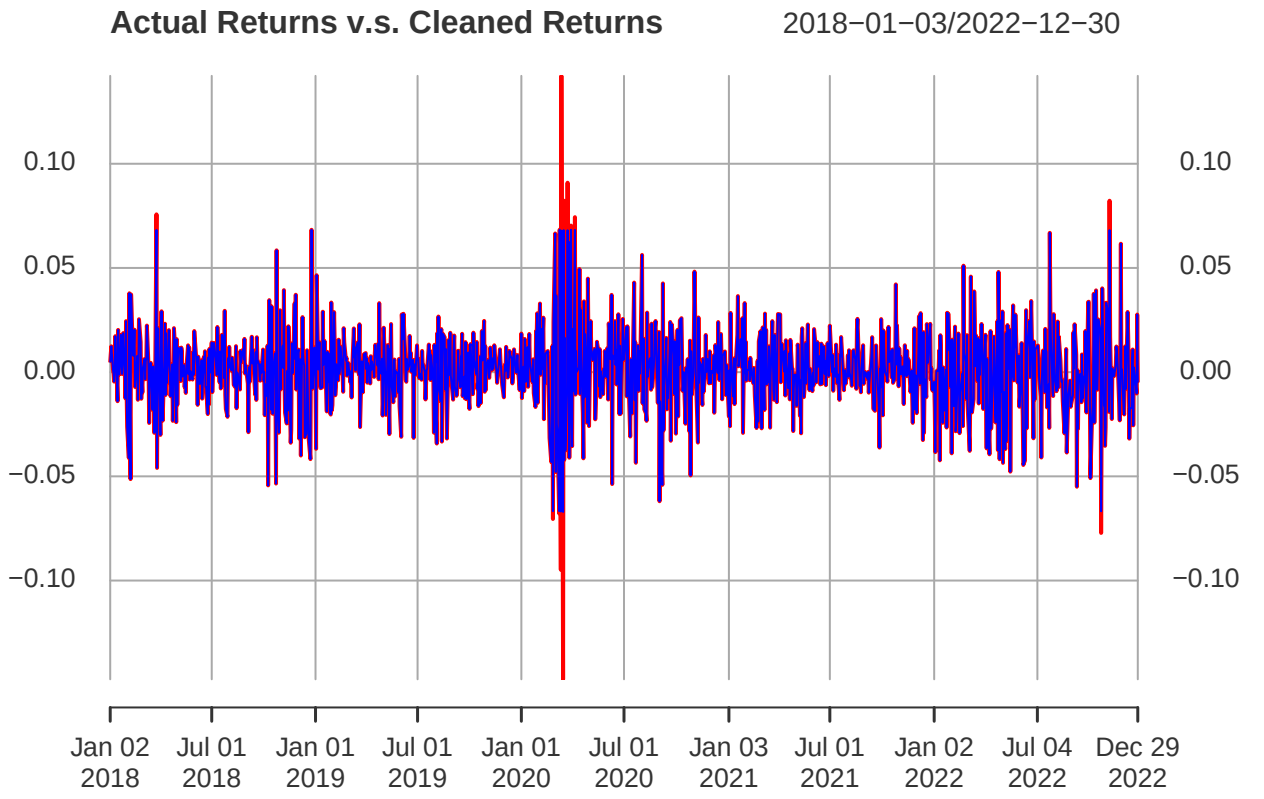

```r
# Compute the average of volatility prediction
aveg_sigma <- xts(rowMeans(model_sigma),
                  order.by = time(model_sigma))
plot(aveg_sigma, main = "Average of Volatility Prediction")
```



Avoid Outliers Distort the Volatility Predictions

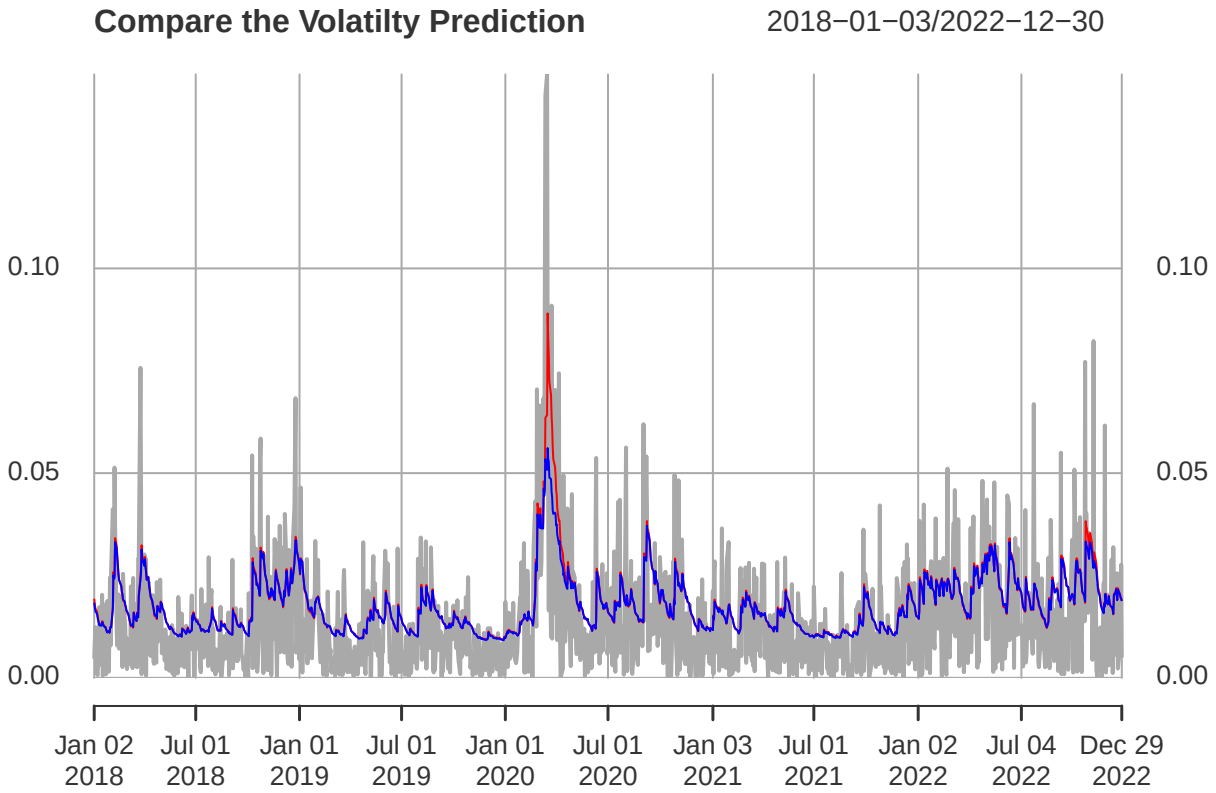
It is useful to reduce the magnitude of the return to an acceptable level, which is also known as winsorization.

```
clean_returns <- Return.clean(returns, method = "boudt")

returnsplot <- plot(returns,
                    col = "red",
                    main = "Actual Returns v.s. Cleaned Returns")
returnsplot <- addSeries(clean_returns,
                         col = "blue",
                         on = 1)
plot(returnsplot)
```



This plots shows the cleaned returns preserving the volatility clusters while winsorizied the extreme outliers in Covid year.

```
clean_garchfit <- ugarchfit(data = clean_returns, spec = garchspec_7)

volplot <- plot(abs(returns),
                col = "darkgray",
                main = "Compare the Volatilty Prediction")
volplot <- addSeries(garch_vol_7,
                     col = "red",
                     on = 1)
volplot <- addSeries(sigma(clean_garchfit),
                     col = "blue",
                     on = 1)
volplot
```



This plot supports the preservation of the volatility clusters of the model, and hence to display the impact of cleaning the data.

Conclusion

In summary, our study reveals that the GARCH model demonstrates enhanced performance by altering the conditional distribution, assumed mean, and variance models. The adept selection of optimal settings further contributes to this improvement. Notably, the model offers valuable insights for decision-making, presenting tools such as VaR plots and simulations. Our findings affirm the GARCH model as a powerful instrument for predicting volatilities and extracting meaningful insights from the data.

Limitations & Recommendations

- 1. GARCH model may have limitations in accounting for external factors that can influence financial markets, such as economic events or sudden market shocks.
  - Incorporating the external variables into the model such as economic indicators, news sentiment scores, or other external factors that might impact volatility. It leads to a multivariate model study.

References

Coghlan, A. n.d. "Using R for Time Series Analysis — Time Series 0.2 Documentation." https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html.

Engle, Robert F, and Victor K Ng. 1993. "Measuring and Testing the Impact of News on Volatility." *J. Finance* 48 (5): 1749–78.

Esprabens, Jake, Ari Arango, and Joshua Kim. 2020. "Chapter 1 Introduction to Time Series." https://bookdown.org/JakeEsprabens/431-Time-Series/introduction-to-time-series.html.

"Exponential Smoothing in R Programming." 2020. https://www.geeksforgeeks.org/exponential-smoothing-in-r-programming/.

Franco, Daniel. 2022. "A Visual Guide to Time Series Decomposition Analysis." https://www.encora.com/insights/a-visual-guide-to-time-series-decomposition-analysis.

"GARCH Volatility Forecasts – Real Options Valuation." n.d. https://rovdownloads.com/blog/garch-volatility-forecasts.

Hayes, Adam. 2005. "What Is Closing Price? Definition, How It's Used, and Example." https://www.investopedia.com/terms/c/closingprice.asp.

———. 2010. "Autoregressive Integrated Moving Average (ARIMA) Prediction Model." https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp.

Hyndman, R. J., & Athanasopoulos, G. 2018. "Chapter 8 ARIMA Models." https://otexts.com/fpp2/arima.html.

Kuhe, D. A. 2019. "Model Order Selection for Arma-GARCH Models Using Information Criteria." https://www.researchgate.net/figure/Model-Order-Selection-for-ARMA-GARCH-Models-using-Information-Criteria_tbl4_337673338.

Kumari, Kajal. 2023. "How to Build Your Time Series Model?" https://www.analyticsvidhya.com/blog/2023/02/how-to-build-your-time-series-model/.

Mattis, Brian. 2021. "Time Series Forecasting in R with Holt-Winters." https://towardsdatascience.com/time-series-forecasting-in-r-with-holt-winters-16ef9ebdb6c0.

Paradkar, M., & Thakar, C. 2023. "Forecasting Stock Prices Using ARIMA Model." https://blog.quantinsti.com/forecasting-stock-returns-using-arima-model/.

Prabhakaran, Selva. n.d. "Time Series Analysis." http://r-statistics.co/Time-Series-Analysis-With-R.html.

Rich, K. T. n.d. "RPubs - Time Series in r: Stationarity Testing." https://rpubs.com/richkt/269797.

Rodrigo, Joaquin Amat. n.d. "Backtesting Forecaster - Skforecast Docs." https://joaquinamatrodrigo.github.io/skforecast/0.4.3/notebooks/backtesting.html.

Saha, S., & Bose, S. 2020. "Forecasting of a Time Series (Stock Market) Data in R." https://statwizards.github.io/Forcasting-A-Time-Series-Stock-Market-Data/.

Sorensen, J. Y. 2022. "Backtesting GARCH with VaR (STAT 6180 Final)." https://www.kaggle.com/code/johnyoungsorensen/backtesting-garch-with-var-stat-6180-final; Kaggle.

The Investopedia Team. 2009. "GARCH Model: Definition and Uses in Statistics." https://www.investopedia.com/terms/g/garch.asp.

"Time Series Analysis : Components & EDA." 2023. https://www.kaggle.com/code/shresthapriya/time-series-analysis-components-eda; Kaggle.

"Time Series: Interpreting ACF and PACF." 2022. https://www.kaggle.com/code/iamleonie/time-series-interpreting-acf-and-pacf; Kaggle.

Tsafack, Idriss. 2021. "GARCH Models with R Programming : A Practical Example with TESLA Stock." https://www.idrisstsafack.com/post/garch-models-with-r-programming-a-practical-example-with-tesla-stock.

Vaidya, D. n.d. "Time Series - Definition, Analysis, Forecasting, Components." https://www.wallstreetmojo.com/time-series/.

"Why Log Returns." 2011. https://quantivity.wordpress.com/2011/02/21/why-log-returns/.

Zvornicanin, E. 2023. "Choosing the Best Q and P from ACF and PACF Plots in Arma-Type Modeling." https://www.baeldung.com/cs/acf-pacf-plots-arma-modeling.