Forecasting a Time Series

Yinan Zhou

9th June 2024

Table of Contents

Introduction

In this project, I applied forecasting methods to predict future values of one or more time series. Additionally, I utilized linear algebra techniques for simple or multiple regression analysis. Finally, I employed one or more error norms introduced to evaluate the accuracies of our forecasts.

Here's a preview of the dataset I utilized in this project:

```
> data <- read_excel("ALY6050_Module3Project_Data.xlsx")
> head(data)
# A tibble: 6 × 6
  Date                Period `AAPL (Apple Inc) / $` `AAPL (Apple Inc) / Volume` `HON (Honeywell Inc) / $` `HON (Honeywell Inc) Volume`
  <dttm>               <dbl>                  <dbl>                       <dbl>                     <dbl>                        <dbl>
1 2019-11-08 00:00:00      1                   64.0                    69986400                      177.                      1636800
2 2019-11-11 00:00:00      2                   64.5                    81821200                      177.                      1594600
3 2019-11-12 00:00:00      3                   64.4                    87388800                      178.                      1816900
4 2019-11-13 00:00:00      4                   65.0                   102734400                      178.                      1855200
5 2019-11-14 00:00:00      5                   64.6                    89182800                      176.                      2208300
6 2019-11-15 00:00:00      6                   65.3                   100206400                      178.                      3242700
```

Here's descriptive statistics for the dataset:

```
> summary(data)
      Date                         Period    AAPL (Apple Inc) / $ AAPL (Apple Inc) / Volume HON (Honeywell Inc) / $
 Min.   :2019-11-08 00:00:00.00   Min.   :  1   Min.   : 55.29       Min.   : 28803800        Min.   :102.5
 1st Qu.:2020-02-12 00:00:00.00   1st Qu.: 65   1st Qu.: 69.77       1st Qu.:106194700        1st Qu.:144.1
 Median :2020-05-14 00:00:00.00   Median :129   Median : 79.12       Median :137265600        Median :162.0
 Mean   :2020-05-14 04:34:33.15   Mean   :129   Mean   : 86.63       Mean   :154854435        Mean   :157.7
 3rd Qu.:2020-08-14 00:00:00.00   3rd Qu.:193   3rd Qu.:108.80       3rd Qu.:185972300        3rd Qu.:173.2
 Max.   :2020-11-13 00:00:00.00   Max.   :257   Max.   :133.95       Max.   :426884800        Max.   :184.3
                                                NA's   :5            NA's   :5                NA's   :5

 HON (Honeywell Inc) Volume
 Min.   :  625500
 1st Qu.: 2369825
 Median : 2977900
 Mean   : 3378529
 3rd Qu.: 4036550
 Max.   :13011800
 NA's   :5
```

Preliminary Work

Firstly, I extracted the 3rd and 5th columns from the data frame, presumably representing the prices of AAPL and HON, respectively, and assigned them to “AAPL_price” and “HON_price”. Then, I converted the data frames “AAPL_price” and “HON_price” into vectors for further analysis. After creating a new data frame “df” containing these two vectors, I removed any rows with missing values using “na.omit(df)”. Finally, I generated a sequence “n_days” that corresponds to the number of rows in “df”, representing the sequence of days.

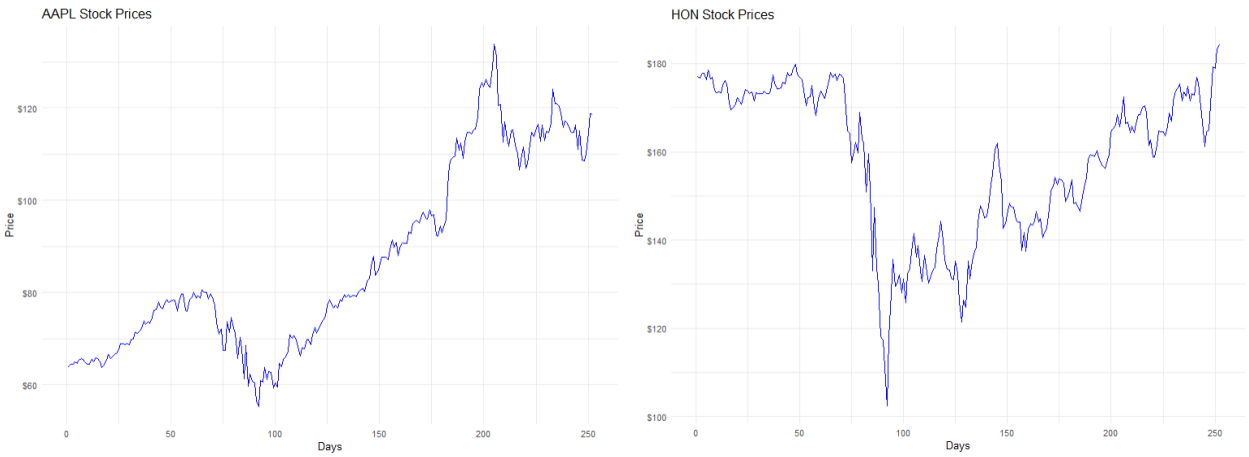
```
> AAPL_price <- data[,3]
> HON_price <- data[,5]
>
> # Transfer the data frames into Vectors
> AAPL_price <-AAPL_price[[1]]
> HON_price <-HON_price[[1]]
>
> df <- data.frame(AAPL_price, HON_price)
> df <- na.omit(df)
> n_days <- seq_along(df[,1])
```

Part 1: Short-term Forecasting

(i) Use a simple line plot of both time series to detect seasonal, irregular, or trend behaviors if any. Write a summary of your observations of both time series in your report.

I created two line plots with the number of days as x-axis and "AAPL_price" and "HON_price" as y-axis.

```
> # (i) Create line plots with the number of days as x-axis and AAPL_price and HON_price as y-axis
> ggplot(df, aes(x = n_days, y = df[,1])) +
+   geom_line(color = "blue", linetype = "solid") +
+   labs(title = "AAPL Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
>
> ggplot(df, aes(x = n_days, y = df[,2])) +
+   geom_line(color = "blue", linetype = "solid") +
+   labs(title = "HON Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```



The left plot depicts the AAPL stock price over time, revealing recurring patterns at regular intervals and a sustained upward trend beyond short-term fluctuations. Additionally, there are unexpected spikes or drops in the data that deviate from the overall trend or seasonal pattern.

On the right, the plot displays the HON stock price over time, showing repeated patterns at regular intervals. However, there is no clear long-term upward or downward movement beyond short-term

fluctuations. Similar to AAPL, there are unexpected spikes or drops in the data that do not adhere to the overall trend or seasonal pattern.

(ii) Perform exponential smoothing to forecast both prices for period 253. Use successive values of 0.15, 0,35, 0.55, and 0.75 for the smoothing parameter α. Next, calculate the MAPD (Mean Absolute Percentage Deviation) of each forecast; and based on the MAPDs, determine the value of α that has yielded the most accurate forecast for each stock. In your report, describe your results; and explain why in your opinion such values of α have yielded the most accurate forecasts for the two stocks.

Initially, I defined two functions as follows:

```
> exponential_smoothing_calculate_next <-function (alpha, Dt, Ft){
+   F_next<-alpha*Dt+(1-alpha)*Ft
+   return (F_next)
+ }
>
> MAPD <- function (observed_vector, predicted_vector){
+   diff <- abs(observed_vector-predicted_vector)
+   return (sum(diff)/sum(observed_vector))
+ }
```

The first function is Exponential Smoothing Function, which calculates the next forecasted value (F_next) using the current observed value (Dt), the current forecasted value (Ft), and the smoothing parameter (alpha). The second function is MAPD Calculation Function, which calculates the Mean Absolute Percentage Deviation (MAPD) between observed and predicted values.

Then, I applied exponential smoothing, and evaluated performance using different alpha values for forecasting AAPL stock prices.

I initialized variables such as the first actual price “D1”, setting the initial forecast value “F1” to match “D1”, defining a list of alpha values (alpha_list) to be tested, and creating an empty list (aapl_predicted_lists) to store predicted values for each alpha.

```
> # AAPL
> # Initialization
> D1 <- AAPL_price[1] # The first actual price in the AAPL_price series
> F1 <- D1 # Initialize the first forecast value to be the same as the first actual price
> alpha_list <- c(0.15, 0.35, 0.55, 0.75)
> aapl_predicted_lists <- list() # Initialize an empty list to store predicted values for each alpha
```

The primary loop iterated through each alpha in the "alpha_list". Within this loop, it initialized the predicted values using F1 and then proceeded to iterate through each day for computing the predicted values using exponential smoothing. This process utilized the "exponential_smoothing_calculate_next" function. After each iteration, the predicted values were stored in "aapl_predicted_lists", and the Mean Absolute Percentage Deviation (MAPD) was computed by comparing the actual prices with the predicted prices. The MAPD for each alpha was printed to assess the effectiveness of exponential smoothing with varying smoothing factors.

```
> # Loop Through Each Alpha
> for (alpha in alpha_list){
+   predicted <- c(F1)
+   for (i in 1:length(n_days)){
+     currentD <- AAPL_price[i]
+     currentF <- predicted[i]
+     predicted<-c(predicted, exponential_smoothing_calculate_next(alpha, currentD, currentF))
+   }
+   aapl_predicted_lists[[as.character(alpha)]] <- predicted # Store the predicted values for the current alpha
+   mapd_aapl_alpha <- MAPD(AAPL_price[1:length(n_days)], predicted[0:length(n_days)])  # Calculate MAPD for the current alpha
+   cat("AAPL MAPD of alpha =", alpha, ":", mapd_aapl_alpha, "\n")
+ }
AAPL MAPD of alpha = 0.15 : 0.03907008
AAPL MAPD of alpha = 0.35 : 0.02483443
AAPL MAPD of alpha = 0.55 : 0.02079728
AAPL MAPD of alpha = 0.75 : 0.01962926
```

After the main loop, there was another loop that checked whether there were at least 253 predicted values for each alpha. If there were, it printed the predicted value for day 253; otherwise, it indicated that there was no 253rd predicted value for that alpha. This check was essential for verifying the completeness of predictions and gaining insight into the forecast accuracy at a specific future point (day 253 in this instance).

```
> # Print the Predicted Value for Day 253
> for (alpha in alpha_list) {
+   predicted_values <- aapl_predicted_lists[[as.character(alpha)]]
+   if (length(predicted_values) >= 253) {
+     cat("Alpha =", alpha, "AAPL Predicted value at day 253:", predicted_values[253], "\n")
+   } else {
+     cat("Alpha =", alpha, "does not have a 253rd predicted value\n")
+   }
+ }
Alpha = 0.15 AAPL Predicted value at day 253: 114.8753
Alpha = 0.35 AAPL Predicted value at day 253: 115.975
Alpha = 0.55 AAPL Predicted value at day 253: 117.4953
Alpha = 0.75 AAPL Predicted value at day 253: 118.3938
```

Based on the provided code and results, the lowest AAPL MAPD value (1.96%) occurred when Alpha was set to 0.75, indicating that this Alpha value resulted in the most accurate forecasts. The detailed results are as follows:

- Alpha = 0.15: AAPL MAPD = 0.03907008
- Alpha = 0.35: AAPL MAPD = 0.02483443
- Alpha = 0.55: AAPL MAPD = 0.02079728
- Alpha = 0.75: AAPL MAPD = 0.01962926

Therefore, considering the MAPD values, it can be concluded that an Alpha value of 0.75 led to the most accurate forecast results. This could be attributed to the fact that larger Alpha values are often more effective in capturing long-term trends in the data, thereby aiding in more precise predictions of changes in AAPL stock prices.

Similarly, I applied exponential smoothing with different alphas to forecast HON stock prices and assessed the forecast accuracy using MAPD as follows:

```
> # HON
> # Initialization
> D1 <- HON_price[1]
> F1 <- D1
> alpha_list <- c(0.15, 0.35, 0.55, 0.75)
> hon_predicted_lists <- list()
>
> # Loop Through Each Alpha
> alpha_list <- c(0.15, 0.35, 0.55, 0.75)
> hon_predicted_lists <- list()
> for (alpha in alpha_list){
+   predicted <- c(F1)
+   for (i in 1:length(n_days)){
+     currentD <- HON_price[i]
+     currentF <- predicted[i]
+     predicted<-c(predicted, exponential_smoothing_calculate_next(alpha, currentD, currentF))
+   }
+   hon_predicted_lists[[as.character(alpha)]] <- predicted
+   mapd_hon_alpha <- MAPD(HON_price[1:length(n_days)], predicted[0:length(n_days)])
+   cat("HON MAPD of alpha =", alpha, ":", mapd_hon_alpha, "\n")
+ }
HON MAPD of alpha = 0.15 : 0.02842749
HON MAPD of alpha = 0.35 : 0.02188635
HON MAPD of alpha = 0.55 : 0.01897397
HON MAPD of alpha = 0.75 : 0.01770951

> # Print the Predicted Value for Day 253
> for (alpha in alpha_list) {
+   predicted_values <- hon_predicted_lists[[as.character(alpha)]]
+   if (length(predicted_values) >= 253) {
+     cat("Alpha =", alpha, "HON Predicted value at day 253:", predicted_values[253], "\n")
+   } else {
+     cat("Alpha =", alpha, "does not have a 253rd predicted value\n")
+   }
+ }
Alpha = 0.15 HON Predicted value at day 253: 175.3274
Alpha = 0.35 HON Predicted value at day 253: 179.9735
Alpha = 0.55 HON Predicted value at day 253: 182.5755
Alpha = 0.75 HON Predicted value at day 253: 183.7235
```

Based on the provided code and results, it is observed that the HON MAPD value is lowest (1.77%) when Alpha is set to 0.75, indicating that an Alpha value of 0.75 has generated the most accurate forecasts. The specific results are outlined below:

- Alpha = 0.15: HON MAPD = 0.02842749
- Alpha = 0.35: HON MAPD = 0.02188635
- Alpha = 0.55: HON MAPD = 0.01897397
- Alpha = 0.75: HON MAPD = 0.01770951

Therefore, based on the MAPD values, it can be concluded that an Alpha value of 0.75 has delivered the most accurate forecast results for HON stock prices. This effectiveness may stem from the fact that larger Alpha values are at times more adept at capturing long-term trends in the data, thereby enhancing the precision of predictions for changes in HON stock prices.

(iii) Use your exponential smoothing forecast of part (ii) with $\alpha = 0.55$ and perform an adjusted exponential smoothing to forecast both prices for period 253. Use successive values of 0.15, 0.25, 0.45,

and 0.85 for the trend parameters β for both stocks. Next, calculate the MAPEs (Mean Absolute Percentage Error) of your forecasts and determine the values of β that have provided the most accurate forecasts for both stocks. In your report, describe your results and explain why, in your opinion, such values of β have yielded the most accurate forecasts.

I first defined the "cal_trend" function to compute the trend component necessary for adjusted exponential smoothing. This function accepts parameters such as beta, representing the trend parameter, Ft for the current forecasted value, "Ft_1" for the previous forecasted value, and "Tt_1" for the previous trend value. The formula utilized within this function is "Tt = beta * (Ft - Ft_1) + (1 - beta) * Tt_1", which calculates the trend value "Tt".

```
> cal_trend <- function (beta, Ft, Ft_1, Tt_1){
+   Tt <- beta*(Ft-Ft_1)+(1-beta)*Tt_1
+   return (Tt)
+ }
```

Secondly, I designed the "MAPE" function to calculate the Mean Absolute Percentage Error between observed and predicted values. This function takes two vectors as inputs: "observed_vector" containing the actual observed values and "predicted_vector" containing the predicted values. The MAPE formula is "MAPE = sum (abs(observed_vector - predicted_vector) / observed_vector) / length(observed_vector)", which computes the average percentage error between observed and predicted values.

```
> MAPE <- function (observed_vector, predicted_vector){
+   diff <- abs(observed_vector-predicted_vector)/observed_vector
+   return (sum(diff)/length(observed_vector))
+ }
```

For forecasting AAPL stock prices, I implemented adjusted exponential smoothing using various beta values and assessed forecast accuracy using Mean Absolute Percentage Error (MAPE).

Initially, I obtained the predicted values for AAPL stock with an alpha value of 0.55 from an existing list named "aapl_predicted_lists". This data was then converted into a usable format by extracting the values from the list. Subsequently, I created a list of beta values [0.15, 0.25, 0.45, 0.85] and initialized an empty list called "aapl_adjusted_lists" to store the adjusted predicted values for AAPL stock calculated using different beta values.

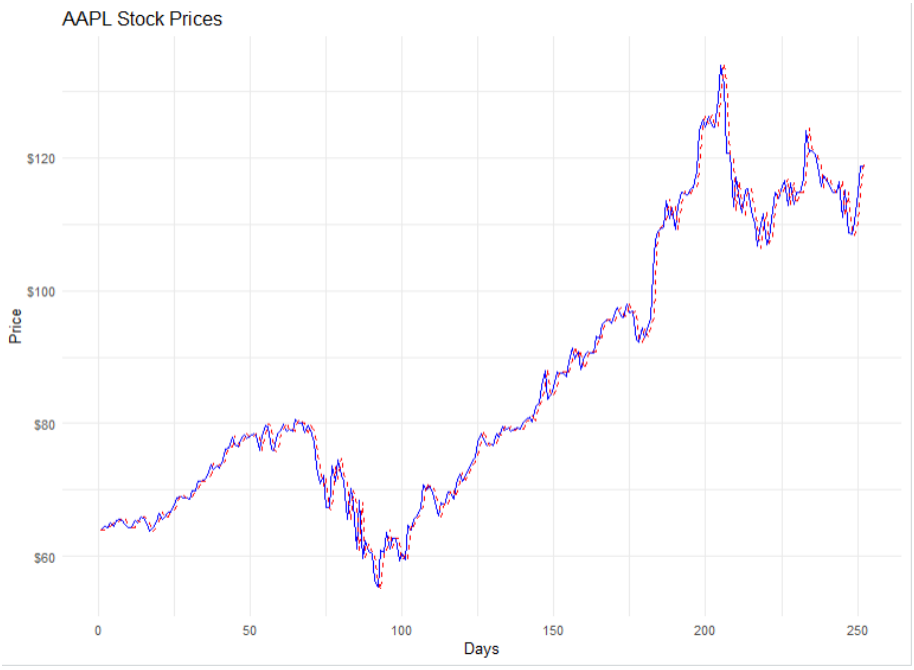
```
> # AAPL
> predicted <- aapl_predicted_lists['0.55']
> predicted <- predicted[[1]] # change data to value
> beta_list <- c(0.15, 0.25, 0.45, 0.85)
> aapl_adjusted_lists <- list()
```

Then, the main loop then iterated through the beta values [0.15, 0.25, 0.45, 0.85], computed trends, adjusted predictions, and stored them along with MAPE values.

```
> for (beta in beta_list){
+   trend <- c(0)
+   for (i in 1:length(n_days)){
+     currentF <- predicted[i]
+     nextF <- predicted[i+1]
+     currentT <- trend[i]
+     trend<-c(trend, cal_trend(beta, nextF, currentF, currentT))
+   }
+   adjusted_predict = predicted + trend
+   aapl_adjusted_lists[[as.character(beta)]] <- adjusted_predict
+   mape_aapl <- MAPE(AAPL_price[1:length(n_days)], adjusted_predict[0:length(n_days)])
+   cat("AAPL MAPE of beta =", beta, ":", mape_aapl, "\n")
+ }
AAPL MAPE of beta = 0.15 : 0.01964077
AAPL MAPE of beta = 0.25 : 0.01944115
AAPL MAPE of beta = 0.45 : 0.01933453
AAPL MAPE of beta = 0.85 : 0.01975939
```

Finally, I printed predicted values for day 253 for each beta and generated a comparison plot between actual AAPL prices and the most accurate adjusted predictions (β=0.45) for visualization purposes.

```
> # Print the Predicted Value for Day 253
> for (beta in beta_list) {
+   adjusted_predict <- aapl_adjusted_lists[[as.character(beta)]]
+   if (length(adjusted_predict) >= 253) {
+     cat("Beta =", beta, "AAPL Predicted value at day 253:", adjusted_predict[253], "\n")
+   } else {
+     cat("Beta =", beta, "does not have a 253rd predicted value\n")
+   }
+ }
Beta = 0.15 AAPL Predicted value at day 253: 118.0357
Beta = 0.25 AAPL Predicted value at day 253: 118.5405
Beta = 0.45 AAPL Predicted value at day 253: 119.2378
Beta = 0.85 AAPL Predicted value at day 253: 119.2209
>
> aapl_predicted_beta4 <- aapl_adjusted_lists['0.85']
> aapl_predicted_beta4 <-aapl_predicted_beta4[[1]]
> ggplot(df, aes(x = n_days, y = df[,1])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=aapl_predicted_beta4[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "AAPL Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```

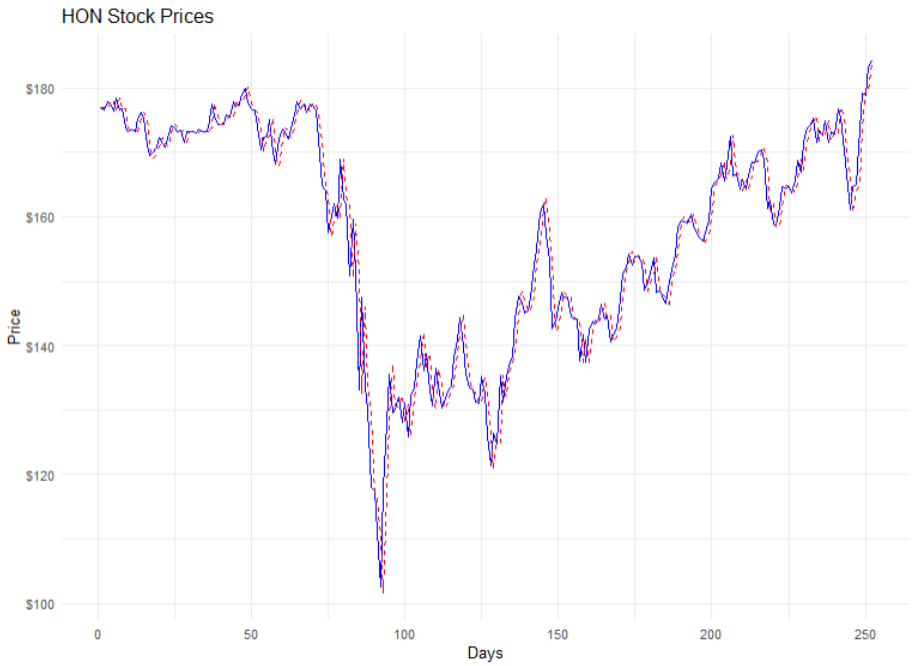


The outcomes of adjusted exponential smoothing for AAPL stock using various beta values (0.15, 0.25, 0.45, 0.85) exhibit differing Mean Absolute Percentage Errors (MAPE). A lower MAPE indicates a more precise forecast. In this instance, the MAPE decreases with increasing beta values, implying that higher beta values result in more accurate forecasts.

This trend can be explained by the essence of exponential smoothing. A higher beta assigns greater significance to recent observations, which may better reflect the genuine underlying trend in stock prices. Consequently, beta values closer to 1 (like 0.85 in this scenario) might produce the most accurate forecasts as they swiftly adapt to data changes, capturing the underlying trend more effectively.

Likewise, I conducted adjusted exponential smoothing for HON stock using different beta values, assessed forecast accuracy via MAPE, and depicted the most precise forecasted values against actual stock prices.

```
> # HON
> predicted <- hon_predicted_lists['0.55']
> predicted <- predicted[[1]]
> beta_list <- c(0.15, 0.25, 0.45, 0.85)
> hon_adjusted_lists <- list()
>
> for (beta in beta_list){
+   trend <- c(0)
+   for (i in 1:length(n_days)){
+     currentF <- predicted[i]
+     nextF <- predicted[i+1]
+     currentT <- trend[i]
+     trend<-c(trend, cal_trend(beta, nextF, currentF, currentT))
+   }
+   adjusted_predict = predicted + trend
+   hon_adjusted_lists[[as.character(beta)]] <- adjusted_predict
+   mape_hon <- MAPE(HON_price[1:length(n_days)], adjusted_predict[0:length(n_days)])
+   cat("HON MAPE of beta =", beta, ":", mape_hon, "\n")
+ }
HON MAPE of beta = 0.15 : 0.01970296
HON MAPE of beta = 0.25 : 0.0193471
HON MAPE of beta = 0.45 : 0.01880899
HON MAPE of beta = 0.85 : 0.01838045
> # Print the Predicted Value for Day 253
> for (beta in beta_list) {
+   adjusted_predict <- hon_adjusted_lists[[as.character(beta)]]
+   if (length(adjusted_predict) >= 253) {
+     cat("Beta =", beta, "AAPL Predicted value at day 253:", adjusted_predict[253], "\n")
+   } else {
+     cat("Beta =", beta, "does not have a 253rd predicted value\n")
+   }
+ }
Beta = 0.15 AAPL Predicted value at day 253: 184.0704
Beta = 0.25 AAPL Predicted value at day 253: 184.7222
Beta = 0.45 AAPL Predicted value at day 253: 185.1923
Beta = 0.85 AAPL Predicted value at day 253: 184.8276
> hon_predicted_beta4 <- hon_adjusted_lists['0.85']
> hon_predicted_beta4 <-hon_predicted_beta4[[1]]
> ggplot(df, aes(x = n_days, y = df[,2])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=hon_predicted_beta4[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "HON Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```



The outcomes of adjusted exponential smoothing for HON stock using different beta values (0.15, 0.25, 0.45, 0.85) illustrate varying Mean Absolute Percentage Errors (MAPE). A lower MAPE indicates a more precise forecast. In this case, as the beta value increases, the MAPE decreases, suggesting that higher beta values result in more accurate forecasts.

This trend can be explained by the essence of exponential smoothing, where a higher beta emphasizes recent observations that likely better reflect the true underlying trend in stock prices. Hence, beta values closer to 1 (e.g., 0.85 in this case) tend to produce the most accurate forecasts by adapting swiftly to data changes and capturing the underlying trend more effectively.

Part 2: Long-term Forecasting

(i) For each stock, use a 3-period weighted moving averages to forecast its value during periods 1 through 100. Use the weights 0.5 (for the most recent period), 0.3 (for the period before the most recent), and 0.2 (for two periods ago). Next, use the observed value for period 101 as the base of a linear trend, and use that linear trend to forecast the values of both stocks for periods 101 through 257. Write a summary of your results in your report. Describe how accurate this method of forecasting has been by comparing the forecasted values for periods 253-257 with their actual "Close" values on those specific days (Hint: check the actual values on https://finance.yahoo.com).

Firstly, I defined a "moving_average" function that takes three input values (x1, x2, x3) and calculates a weighted moving average using coefficients 0.5, 0.3, and 0.2, respectively.

```
> # (i)
> moving_average <-function (x1, x2, x3){
+   return (0.5*x3+0.3*x2+0.2*x1)
+ }
```

Next, I employed a combination of moving average and linear regression techniques to forecast AAPL stock prices.

I initialized “aapl_100” with the first three prices from the “AAPL_price” dataset. A for loop was then used to iterate over the AAPL_price data, starting from the third element (i = 3) up to the 100th element. Inside the loop, x1, x2, and x3 were updated with the current, previous, and second previous prices, respectively, and the “moving_average” function was applied to generate the next predicted price. These predicted prices were appended to the “aapl_100” list.

```
> aapl_100 <- AAPL_price[1:3]
>
> for (i in 3:100){
+   x1<-AAPL_price[i-2]
+   x2<-AAPL_price[i-1]
+   x3<-AAPL_price[i]
+   aapl_100 <- c(aapl_100, moving_average(x1,x2,x3))
+ }
```

Vectors “sample_days” and “sample_AAPL_price” were created to capture data from day 101 to day 252. A linear regression model (lm) was then fitted using “sample_AAPL_price” as the response variable and “sample_days” as the predictor variable. The coefficients of the linear model were obtained, representing the intercept and slope.

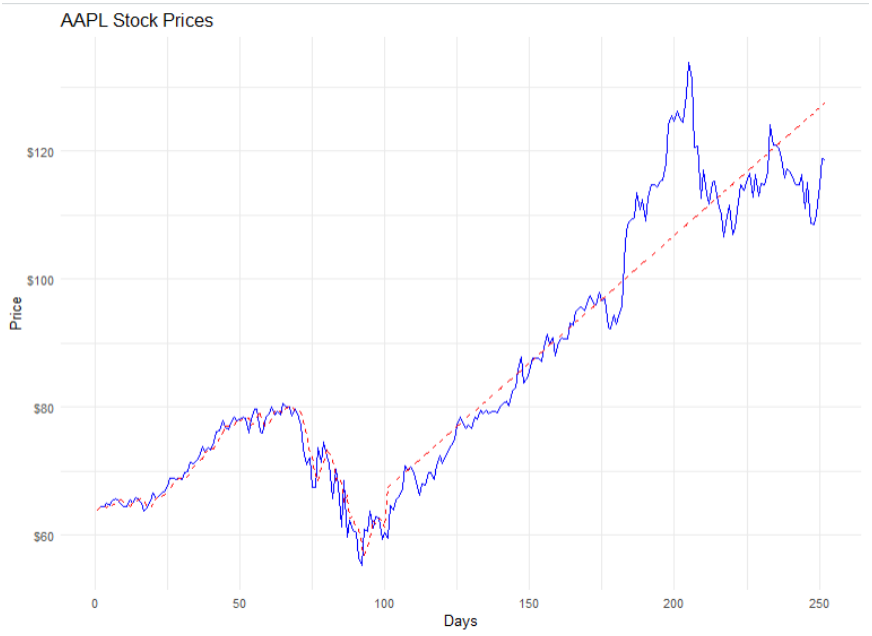
```
> sample_days <- c(101:252)
> sample_AAPL_price <- AAPL_price[101:252]
>
> # Fit a linear model
> model <- lm(sample_AAPL_price ~ sample_days)
> # Get the coefficients
> coefficients <- coef(model)
> coefficients
(Intercept) sample_days
 27.0970290   0.3988899
```

Using these coefficients, trend values were calculated for days 101 to 257, and the “aapl_predicted” vector was formed by combining the first 100 predicted prices (from the moving average) with the predicted trend values.

```
> # Calculate the trend values
> calculate_days <- c(101:257)
> aapl_predicted_trend_values <- coefficients[1] + coefficients[2] * calculate_days
> aapl_predicted <- c(aapl_100[1:100],aapl_predicted_trend_values)
```

Additionally, I utilized “ggplot” to create a plot showing actual AAPL stock prices (df[,1]) and predicted prices (aapl_predicted) against the days. In this plot, the solid blue line represents actual prices, while the dashed red line represents predicted prices. This visualization aids in understanding the performance of adjusted exponential smoothing forecasts compared to actual stock prices.

```
> ggplot(df, aes(x = n_days, y = df[,1])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=aapl_predicted[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "AAPL Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
>
```



Finally, I printed predicted prices and observed prices for periods 253 to 257, highlighting the comparison between the linear trend's predicted prices and actual observed prices.

```
> observed_aapl_price<-c(AAPL_price[1:252], 116.32,115.97,119.49,119.21,119.26)
> for (i in c(253:257)){
+   cat ("period ", i, ": linear trend predicted price = ", aapl_predicted[i], ", observed price = ", observed_aapl_price[i],'\n')
+ }
period  253 : linear trend predicted price =  128.0162 , observed price =  116.32
period  254 : linear trend predicted price =  128.4151 , observed price =  115.97
period  255 : linear trend predicted price =  128.8139 , observed price =  119.49
period  256 : linear trend predicted price =  129.2128 , observed price =  119.21
period  257 : linear trend predicted price =  129.6117 , observed price =  119.26
```

The results and line plot comparison revealed that the forecasted AAPL prices for periods 253 to 257 consistently exceeded the actual AAPL observed prices. Therefore, the linear trend assumption may not accurately capture the complexities and volatilities of AAPL stock prices.

Similarly, I forecasted HON stock prices using a combination of moving average and linear regression techniques as follows:

```
> #HON
> observed_hon_price<-c(HON_price[1:252], 196.99,201.98,199.29,197.24,201.54)
> # (i)
> moving_average <-function (x1, x2, x3){
+   return (0.5*x3+0.3*x2+0.2*x1)
+ }
> hon_100 <- HON_price[1:3]
>
> for (i in 3:100){
+   x1<-HON_price[i-2]
+   x2<-HON_price[i-1]
+   x3<-HON_price[i]
+   hon_100 <- c(hon_100, moving_average(x1,x2,x3))
+ }
>
> sample_days <- c(101:252)
> sample_HON_price <- HON_price[101:252]
>
```

```
> # Fit a linear model
> model <- lm(sample_HON_price ~ sample_days)
> # Get the coefficients
> coefficients <- coef(model)
> coefficients
(Intercept) sample_days
 99.0611053   0.3043024
>
> # Calculate the trend values
> calculate_days <- c(101:257)
> hon_predicted_trend_values <- coefficients[1] + coefficients[2] * calculate_days
> hon_predicted <- c(hon_100[1:100],hon_predicted_trend_values)
>
> ggplot(df, aes(x = n_days, y = df[,2])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=hon_predicted[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "HON Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```


```
> observed_hon_price<-c(HON_price[1:252], 196.99,201.98,199.29,197.24,201.54)
> for (i in c(253:257)){
+   cat ("period ", i, ": linear trend predicted price = ", hon_predicted[i], ", observed price = ", observed_hon_price[i],'\n')
+ }
period  253 : linear trend predicted price =  176.0496 , observed price =  196.99
period  254 : linear trend predicted price =  176.3539 , observed price =  201.98
period  255 : linear trend predicted price =  176.6582 , observed price =  199.29
period  256 : linear trend predicted price =  176.9625 , observed price =  197.24
period  257 : linear trend predicted price =  177.2668 , observed price =  201.54
```

Based on the results and the comparison in the line plot, it's evident that the forecasted HON prices for periods 253 to 257 consistently fall below the actual observed prices of HON stock. This indicates that the linear trend assumption might not adequately capture the complexities and fluctuations in HON stock prices.

(ii) Calculate the MAPEs (Mean Absolute Percentage Error) of your forecasts in question (i) above and compare them with the values obtained for your forecasts in Part 1. For each stock, describe which method has yielded a most accurate forecast.

Firstly, I calculated the MAPEs for the predicted prices of both AAPL and HON using the moving average and linear trend forecast methods. From the result, MAPE for the predicted prices of AAPL is approximately 3.76%, and MAPE for the predicted prices of HON is approximately 2.69%.

```
> # (ii)
> mape_aapl<-MAPE(observed_aapl_price[4:252],aapl_predicted[4:252])
> cat ("AAPL MAPE = ", mape_aapl, '\n')
AAPL MAPE =  0.03758821

> # (ii)
> mape_hon<-MAPE(observed_hon_price[4:252],hon_predicted[4:252])
> cat ("HON MAPE = ", mape_hon, '\n')
HON MAPE =  0.02685031
```

Then, I made comparison of Forecasting Methods:

AAPL Stock:

- Moving Average and Linear Trend: MAPE = 3.76%
- Adjusted Exponential Smoothing (alpha=0.55, beta=0.45): MAPE = 1.93%

HON Stock:

- Moving Average and Linear Trend: MAPE = 2.69%
- Adjusted Exponential Smoothing (alpha=0.55, beta=0.85): MAPE = 1.84%

From the comparison, the exponential smoothing method provided more accurate forecasts for both AAPL and HON stocks, as indicated by significantly lower MAPE values compared to the moving average and linear trend methods. Exponential smoothing captures trends and volatilities more effectively, making it a better choice for stock price predictions.

Part 3: Regression

(i) For each stock, use simple regression of stock values versus the time periods to predict its values for periods 1 through 257. In your report, describe how the accuracy of this prediction has been compared to the methods used in Parts 1 and 2 of this project.

For forecasting AAPL stock prices, I used a linear regression model. I prepared the necessary data, set up a sequence of days (sample_days) and extracted corresponding actual AAPL prices for the first 252 days (sample_AAPL_price).

```
> # AAPL
> #(i)
> sample_days <- 1:252
> sample_AAPL_price <- AAPL_price[1:252]
```

Next, I fitted a linear regression model using the “lm” function, where AAPL prices are regressed on the days. This model is used to predict future AAPL prices for the first 257 days (future_days) using the “predict” function.

```
> # Fit a linear model
> model <- lm(sample_AAPL_price ~ sample_days)
> # Get the coefficients
> coefficients <- coef(model)
> coefficients
(Intercept) sample_days
 55.8042734   0.2437129
> # Predict values for periods 1 through 257
> future_days <- 1:257
> aapl_predicted <- predict(model, newdata = data.frame(sample_days = future_days))
```

After obtaining the predicted AAPL prices, I calculated MAPE between the observed AAPL prices and the predicted prices for the first 252 days. This metric helps evaluate the accuracy of the forecasted prices.

```
> mape_aapl<-MAPE(observed_aapl_price[1:252],aapl_predicted[1:252])
> cat ("AAPL MAPE = ", mape_aapl, '\n')
AAPL MAPE =  0.1056076
```

The simple regression method for AAPL stock had a MAPE of approximately 10.56%, significantly higher than the methods in Parts 1 and 2, suggesting less accuracy in predicting stock prices. Therefore, based on the MAPE values and the forecasting accuracy, the simple regression method appears to be less accurate for predicting AAPL stock prices compared to the methods used in Parts 1 and 2.

Finally, I generated a plot using “ggplot”, displaying both the actual AAPL prices (represented by a solid blue line) and the predicted AAPL prices (represented by a dashed red line) against the days. This visualization aids in comparing the forecasted prices with the actual prices.

```
> ggplot(df, aes(x = n_days, y = df[,1])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=aapl_predicted[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "AAPL Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```

From the line plot comparison, while the linear regression model provided a general trend, it may not have fully captured the complexities and volatilities inherent in AAPL stock prices, leading to discrepancies between the predicted and actual values.

Similarly, I used a linear regression model to forecast HON stock prices for period 1 to 257, calculated MAPE between the observed HON prices and the predicted prices, and visualized both the actual HON prices (represented by a solid blue line) and the predicted HON prices (represented by a dashed red line) against the days for comparison.

```
> # Fit a linear model
> model <- lm(sample_HON_price ~ sample_days)
> # Get the coefficients
> coefficients <- coef(model)
> coefficients
 (Intercept)  sample_days
161.64969350  -0.03138926
>
> # Predict values for periods 1 through 257
> future_days <- 1:257
> hon_predicted <- predict(model, newdata = data.frame(sample_days = future_days))
> mape_hon<-MAPE(observed_hon_price[1:252],hon_predicted[1:252])
> cat ("HON MAPE = ", mape_hon, '\n')
HON MAPE =  0.09758568
```

The simple regression method for HON stock had a MAPE of approximately 9.76%, significantly higher than the methods in Parts 1 and 2, suggesting less accuracy in predicting stock prices. Therefore, based on the MAPE values and the forecasting accuracy, the simple regression method appears to be less accurate for predicting HON stock prices compared to the methods used in Parts 1 and 2.

Also, I generated a plot using “ggplot”, displaying both the actual HON prices (represented by a solid blue line) and the predicted HON prices (represented by a dashed red line) against the days. This visualization aids in comparing the forecasted prices with the actual prices.

```
> ggplot(df, aes(x = n_days, y = df[,2])) +
+   geom_line(color = "blue", linetype = "solid") +
+   geom_line(aes(y=hon_predicted[1:252]), color = "red", linetype = "dashed") +
+   labs(title = "HON Stock Prices",
+        x = "Days",
+        y = "Price") +
+   scale_y_continuous(labels = scales::dollar) +
+   theme_minimal()
```

From the line plot comparison, while the linear regression model provided a general trend, it may not have fully captured the complexities and volatilities inherent in HON stock prices, leading to discrepancies between the predicted and actual values.

(ii) Perform a residual analysis of your simple regression to verify whether regression is appropriate to use for each of the given data. In particular, determine:

• Whether the residuals are independent

• Whether the residuals are homoscedastic.

• Whether the residuals are normally distributed by plotting a Normal probability plot of the residuals

• Whether the residuals are normally distributed by performing a Chi-squared test for Normality of the residuals.

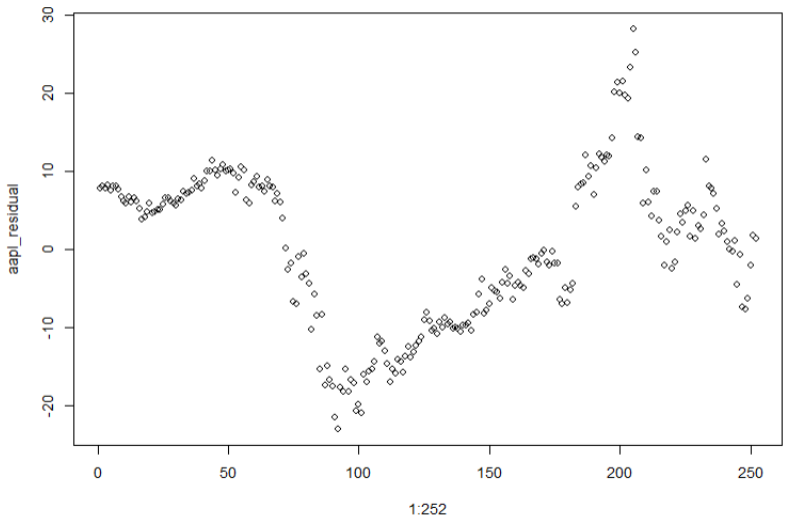
1. APPL

- **Calculate Residuals:**

```
> #(ii)
> aapl_residual <- observed_aapl_price[1:252] - aapl_predicted[1:252]
```

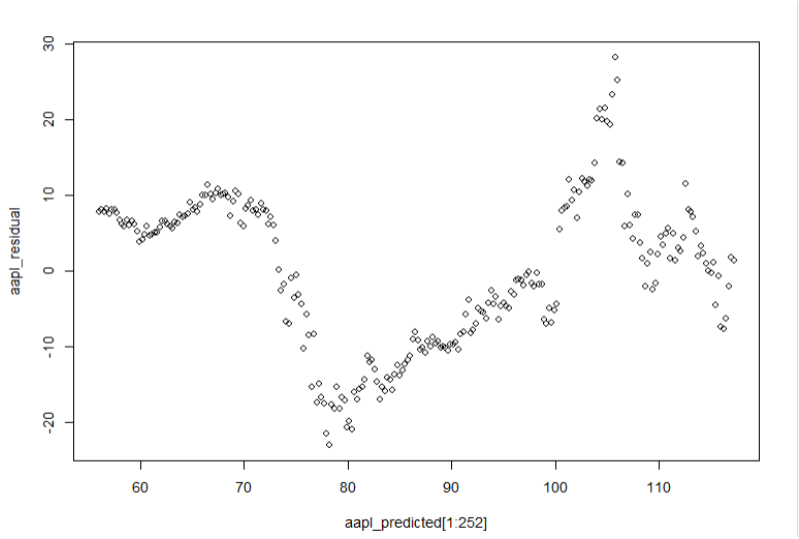
- **Independence Check:** I plotted the residuals against the independent variable (time periods) to check for independence. The plot reveals certain patterns, indicating that the residuals are not independent.

```
> # Independence (plot the residual vs independent variable)
> plot(1:252, aapl_residual)
```



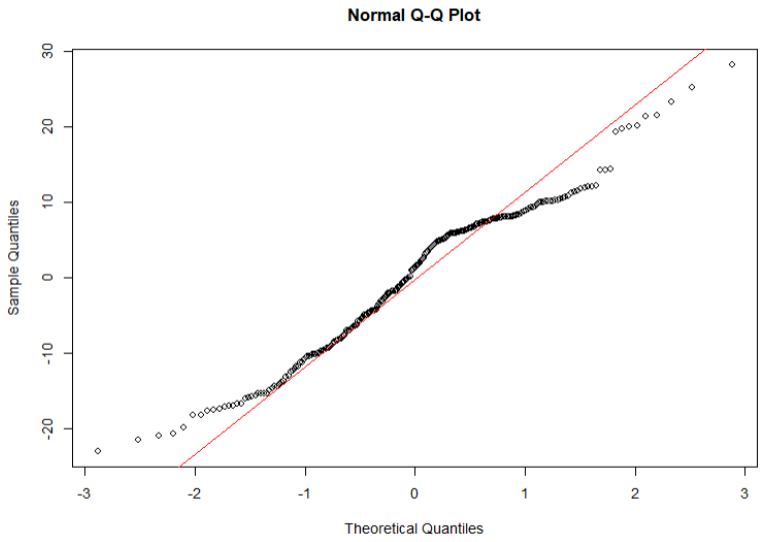
- **Homoscedasticity Check:** I plotted the residuals against the predicted values (y-hat) to check for homoscedasticity. From the plot, we observed certain patterns, indicating that the residuals do not exhibit homoscedasticity.

```
> # Homoscedastic (plot residual vs predicted y)
> plot(aapl_predicted[1:252], aapl_residual)
```



- **Normal Probability Plot:** I created a Normal Q-Q plot of the residuals to visually assess normality. If the residuals lie approximately on the straight line, it suggests that they are normally distributed. However, since we observe a deviation from the straight line, it indicates a departure from normality.

```
> # normal probability plot
> qqnorm(aapl_residual)
> qqline(aapl_residual, col = "red")
```



- **Chi-squared Test:** I conducted a Chi-squared test to assess the normality of the residuals. This involved creating a histogram to visualize the distribution of residuals and calculating observed and expected frequencies based on binning and assuming a normal distribution. Subsequently, I performed the Chi-squared test and printed the results.

```
> # Chi-square test
> # Define the number of bins
> hist(aapl_residual,breaks = 5 )
> num_bins <- 5
> # Create breaks for binning
> breaks <- quantile(aapl_residual, probs = seq(0, 1, length.out = num_bins + 1))
> # Bin the residual
> obs_freq <- table(cut(aapl_residual, breaks = breaks, include.lowest = TRUE))
> # Calculate expected frequencies assuming a normal distribution
> expected_freq <- diff(pnorm(breaks, mean = mean(aapl_residual), sd = sd(aapl_residual))) * length(aapl_residual)
> # Perform the Chi-squared test
> chi_sq_test_aaple <- chisq.test(obs_freq, p = expected_freq / sum(expected_freq))
> # Print the results of the Chi-squared test
> print(chi_sq_test_aaple)

        Chi-squared test for given probabilities

data:  obs_freq
X-squared = 40.424, df = 4, p-value = 3.536e-08
```

Based on the result, the calculated Chi-squared value is 40.424, and the p-value associated with the Chi-squared test is very low, approximately 3.536e-08.

The Chi-squared test assesses the null hypothesis that the residuals conform to a normal distribution. With a p-value significantly lower than the conventional significance level of 0.05, we reject the null hypothesis. The substantial Chi-squared value further reinforces this rejection, indicating a notable deviation from the anticipated normal distribution. Consequently, we can conclude that the residuals do not follow a normal distribution.

Overall, since the residuals are not independent, not homoscedastic, and not normally distributed, regression is not appropriate for predicting AAPL stock values.

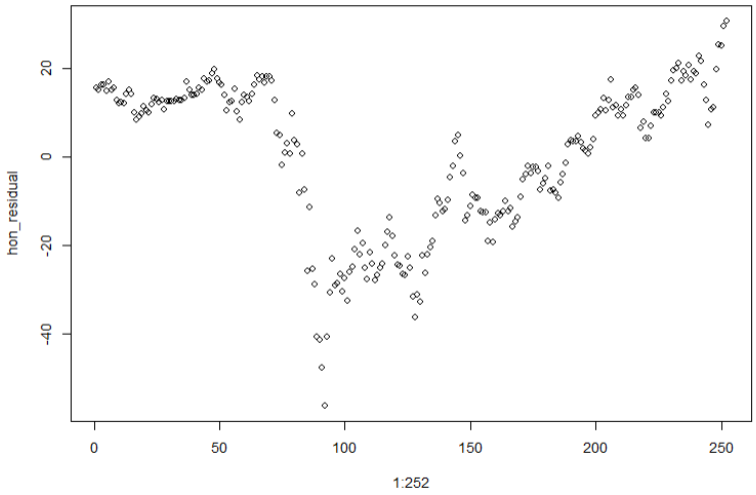
2. HON

- **Calculate Residuals:**

```
> #(ii)
> hon_residual <- observed_hon_price[1:252] - hon_predicted[1:252]
```

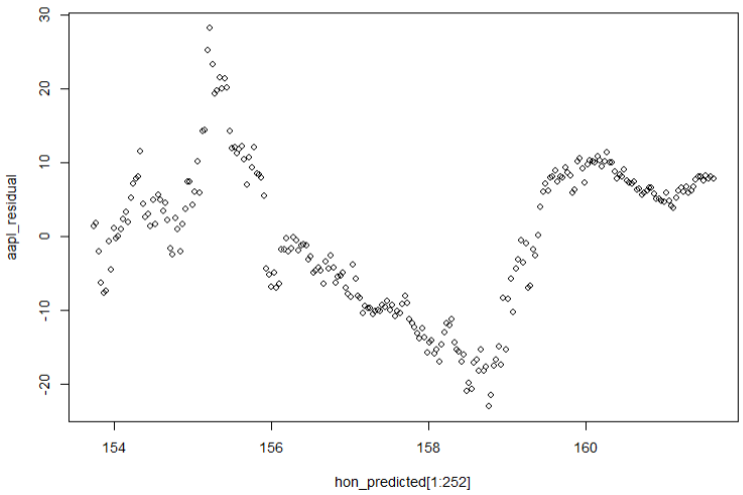
- **Independence Check:** I plotted the residuals against the independent variable (time periods) to check for independence. The plot reveals certain patterns, indicating that the residuals are not independent.

```
> # Independence (plot the residual vs independent variable)
> plot(1:252, hon_residual)
```



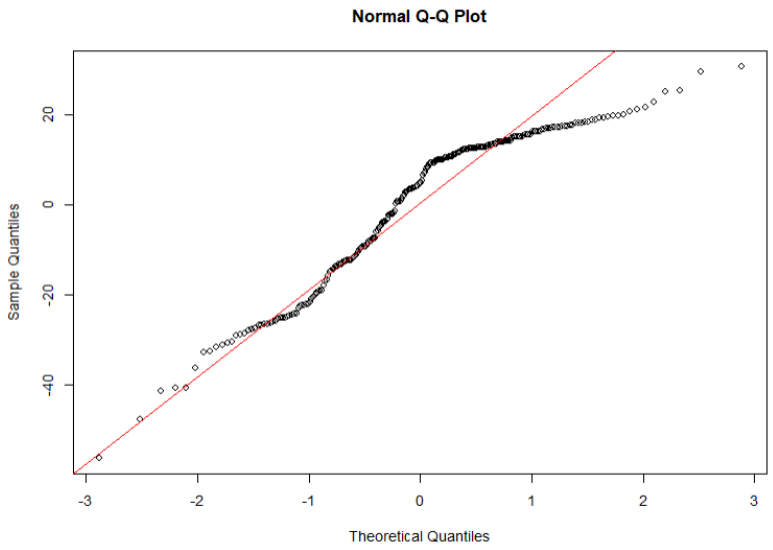
- **Homoscedasticity Check:** I plotted the residuals against the predicted values (y-hat) to check for homoscedasticity. From the plot, we observed certain patterns, indicating that the residuals do not exhibit homoscedasticity.

```
> # Homoscedastic (plot residual vs predicted y)
> plot(hon_predicted[1:252], aapl_residual)
```



- **Normal Probability Plot:** I created a Normal Q-Q plot of the residuals to visually assess normality. If the residuals lie approximately on the straight line, it suggests that they are normally distributed. However, since we observe a deviation from the straight line, it indicates a departure from normality.

```
> # normal probability plot
> qqnorm(hon_residual)
> qqline(hon_residual, col = "red")
```



- **Chi-squared Test:** I conducted a Chi-squared test to assess the normality of the residuals. This involved creating a histogram to visualize the distribution of residuals and calculating observed and expected frequencies based on binning and assuming a normal distribution. Subsequently, I performed the Chi-squared test and printed the results.

```
> # Chi-square test
> # Define the number of bins
> hist(hon_residual,breaks = 5 )
> num_bins <- 5
> # Create breaks for binning
> breaks <- quantile(hon_residual, probs = seq(0, 1, length.out = num_bins + 1))
> # Bin the residual
> obs_freq <- table(cut(hon_residual, breaks = breaks, include.lowest = TRUE))
> # Calculate expected frequencies assuming a normal distribution
> expected_freq <- diff(pnorm(breaks, mean = mean(hon_residual), sd = sd(hon_residual))) * length(hon_residual)
> # Perform the Chi-squared test
> chi_sq_test_hon<- chisq.test(obs_freq, p = expected_freq / sum(expected_freq))
> # Print the results of the Chi-squared test
> print(chi_sq_test_hon)

        Chi-squared test for given probabilities

data:  obs_freq
X-squared = 63.92, df = 4, p-value = 4.345e-13
```

Based on the result, the calculated Chi-squared value is 63.92, and the p-value associated with the Chi-squared test is very low, approximately 4.345e-13.

The Chi-squared test assesses the null hypothesis that the residuals conform to a normal distribution. With a p-value significantly lower than the conventional significance level of 0.05, we reject the null hypothesis. The substantial Chi-squared value further reinforces this rejection, indicating a notable deviation from the anticipated normal distribution. Consequently, we can conclude that the residuals do not follow a normal distribution.

Overall, since the residuals are not independent, not homoscedastic, and not normally distributed, regression is not appropriate for predicting HON stock values.

Question

Suppose that you have decided to form a portfolio Π (Pi) consisting of the above two stock types (denote a share value of AAPL by X and that of HON by Y). You are however undecided as to what percentage of your investment should be allocated to the AAPL shares and what percentage should be allocated to HON shares. Let these percentages be denoted by P and Q respectively (Obviously, P + Q=100%). In your opinion, what are good values to select for P and Q?

1. **Data Preparation:**

- I started by cleaning the dataset titled "data," which is assumed to include historical stock prices for AAPL and HON in 2022, by eliminating rows with missing values.

```
> # Remove rows with NA values
> data <- na.omit(data)
>
> # Extract AAPL and HON stock prices
> aapl_prices <- data[, 3]
> aapl_prices <- aapl_prices[[1]]
> hon_prices <- data[, 5]
> hon_prices <- hon_prices[[1]]
```

2. **Calculating Returns and Risk Measures:**

- Daily returns for AAPL and HON are computed by taking the logarithmic difference of their stock prices. These returns are essential for assessing the historical performance of each stock.

```
> # Calculate daily returns
> aapl_returns <- diff(log(aapl_prices))
> hon_returns <- diff(log(hon_prices))
>
> # Combine returns into a single data frame
> returns <- data.frame(Date = data$Date[-1], AAPL_Returns = aapl_returns, HON_Returns = hon_returns)
```

- Mean daily returns and standard deviations of daily returns are then calculated for both AAPL and HON. These measures provide insights into the average performance and volatility of each stock. In this case, AAPL (Apple) shares have a higher mean daily return compared to HON (Honeywell International Inc.) shares, indicating that AAPL has historically provided higher average daily returns. AAPL shares have a slightly higher standard deviation compared to HON shares, suggesting that AAPL's returns fluctuate more widely on a daily basis.

```
> # Calculate mean daily returns
> mean_aapl_returns <- mean(returns$AAPL_Returns)
> mean_aapl_returns
[1] 0.002463516
> mean_hon_returns <- mean(returns$HON_Returns)
> mean_hon_returns
[1] 0.0001597069
>
> # Calculate standard deviation of daily returns
> sd_aapl_returns <- sd(returns$AAPL_Returns)
> sd_aapl_returns
[1] 0.02909126
> sd_hon_returns <- sd(returns$HON_Returns)
> sd_hon_returns
[1] 0.02798638
```

3. **Sharpe Ratio Calculation:**

- The Sharpe Ratio is computed for both AAPL and HON using the formula “Mean Return – Risk Free Risk) / Standard Deviation”.
- Government bonds are widely acknowledged as some of the safest investment vehicles, and as a result, their interest rates are frequently adopted as the standard for the risk-free rate. Specifically, the three-month Treasury bond rate in the United States, commonly abbreviated as the three-month Treasury yield, can serve as this risk-free rate. This is due to the fact that U.S. Treasury bonds are esteemed as one of the market's safest assets, with their interest rates directly representing the expected return for risk-free investments. (Napoletano, 2022) For the purposes of this example, the risk-free rate is set at 5.25%, which corresponds to the three-month Treasury bond rate in the United States as of June 9th, 2024.
- The Sharpe Ratio serves as a key metric for evaluating risk-adjusted returns. (Fernando,2024) A higher Sharpe Ratio indicates better risk-adjusted performance. In this case, Both AAPL and HON shares have negative Sharpe Ratios in this example, which means that their returns are not compensating for the level of risk taken, given the risk-free rate of 5.25%. However, AAPL shares have a relatively higher Sharpe Ratio compared to HON shares, implying that AAPL may offer a better risk-adjusted return than HON.

```
> # Calculate Sharpe Ratio
> risk_free_rate <- 0.0525  # Use the three-month Treasury bond rate 5.25% in this case
> sharpe_aapl <- (mean_aapl_returns - risk_free_rate) / sd_aapl_returns
> sharpe_aapl
[1] -1.719983
> sharpe_hon <- (mean_hon_returns - risk_free_rate) / sd_hon_returns
> sharpe_hon
[1] -1.870206
```

4. **Optimal Allocation Determination:**

- Based on the Sharpe Ratios, the optimal allocation suggests investing approximately 47.91% of the portfolio in AAPL shares and 52.09% in HON shares. This allocation aims to maximize risk-adjusted returns given the historical performance and risk characteristics of these two stocks.

```
> # Calculate optimal allocation weights
> sharpe_weights <- c(sharpe_aapl, sharpe_hon) / sum(c(sharpe_aapl, sharpe_hon))
> aapl_allocation <- sharpe_weights[1]
> hon_allocation <- sharpe_weights[2]
>
> # Print the results
> print(paste("AAPL allocation percentage:", round(aapl_allocation * 100, 2), "%"))
[1] "AAPL allocation percentage: 47.91 %"
> print(paste("HON allocation percentage:", round(hon_allocation * 100, 2), "%"))
[1] "HON allocation percentage: 52.09 %"
```

References

Fernando, J. (2024, January 30). Sharpe Ratio: Definition, Formula, and Examples. *Investopedia*. https://www.investopedia.com/terms/s/sharperatio.asp

Napoletano, E. (2022, June 28). The Risk-Free Rate. *Forbes Advisor*. https://www.forbes.com/advisor/author/enapoletano/