Forecasting Hermès Stock Price Using Monte Carlo Methods and Time Series Analysis

Sara Mezuri

2023-04-10

Introduction

Hermès is a global luxury fashion brand known for its high-quality products and timeless designs. The company has a strong presence in the market and is valued by investors for its consistent financial performance and strong brand reputation. However, investing in luxury fashion companies can be risky, and investors need to evaluate the potential risks and returns of their investment decisions. To address this challenge, this project aims to use Monte Carlo simulations and time series analysis to forecast the stock prices of Hermès and evaluate different investment strategies.

In this project, we analyze the historical stock prices of Hermès and use time series analysis techniques to identify patterns and trends in the data. We then use the time series model to forecast the future stock prices of Hermès for a given period. Additionally, we conduct Monte Carlo simulations to generate thousands of possible outcomes for the forecasted prices, taking into account various factors that affect the stock prices such as market volatility, industry trends, and economic indicators.

Historical Stock Market Data

We will collect monthly stock prices from Investing.com. (https://www.investing.com/equities/HermÃĺs-international-historical-data)

```
# Read in the data
setwd("C:/Users/saram/Desktop/OU/Winter 2023/STA 5006 (Stat Computing)/Project")
mydata <- read.csv("HRMS Historical Data.csv", header = TRUE)

# Show the first and last six rows of the data
head(mydata)
```

```
##        Date  Price     Open     High      Low   Vol. Change..
## 1  4/1/2023 1909.2 1,856.60 1,918.80 1,845.00 55.48K    2.42%
## 2  3/1/2023 1864.0 1,732.00 1,874.50 1,661.00  1.54M    8.66%
## 3  2/1/2023 1715.5 1,711.00 1,778.00 1,648.50  1.32M    0.06%
## 4  1/1/2023 1714.5 1,461.00 1,717.50 1,450.00  1.29M   18.65%
## 5 12/1/2022 1445.0 1,552.00 1,585.50 1,432.00  1.25M   -6.32%
## 6 11/1/2022 1542.5 1,337.00 1,559.00 1,305.00  1.81M   17.70%
```

```
tail(mydata)
```

```
##          Date Price Open High  Low    Vol. Change..
## 353 12/1/1993  7.96 6.78 8.08 6.61   1.32M   17.23%
## 354 11/1/1993  6.79 6.59 7.03 6.45   1.02M    3.66%
## 355 10/1/1993  6.55 6.69 6.78 6.35   1.27M   -2.82%
## 356  9/1/1993  6.74 5.76 6.83 5.39   1.06M   17.01%
## 357  8/1/1993  5.76 5.34 6.08 5.13 834.47K    7.87%
## 358  7/1/1993  5.34  5.3 5.34 4.95 708.17K    2.10%
```

```r
#Convert the data
mydata$Date<-as.Date(mydata$Date, format="%m/%d/%Y")

# Reverse the order of the data
Price_new <- rev(mydata$Price)
Date_new <- rev(as.Date(mydata$Date))
```

```r
# Create the plot (plot fig.1)
plot(mydata$Date, mydata$Price,
     type = "o", xlab = "Time", ylab = "Price", main = "Hermès Stock Price since 1993")
```

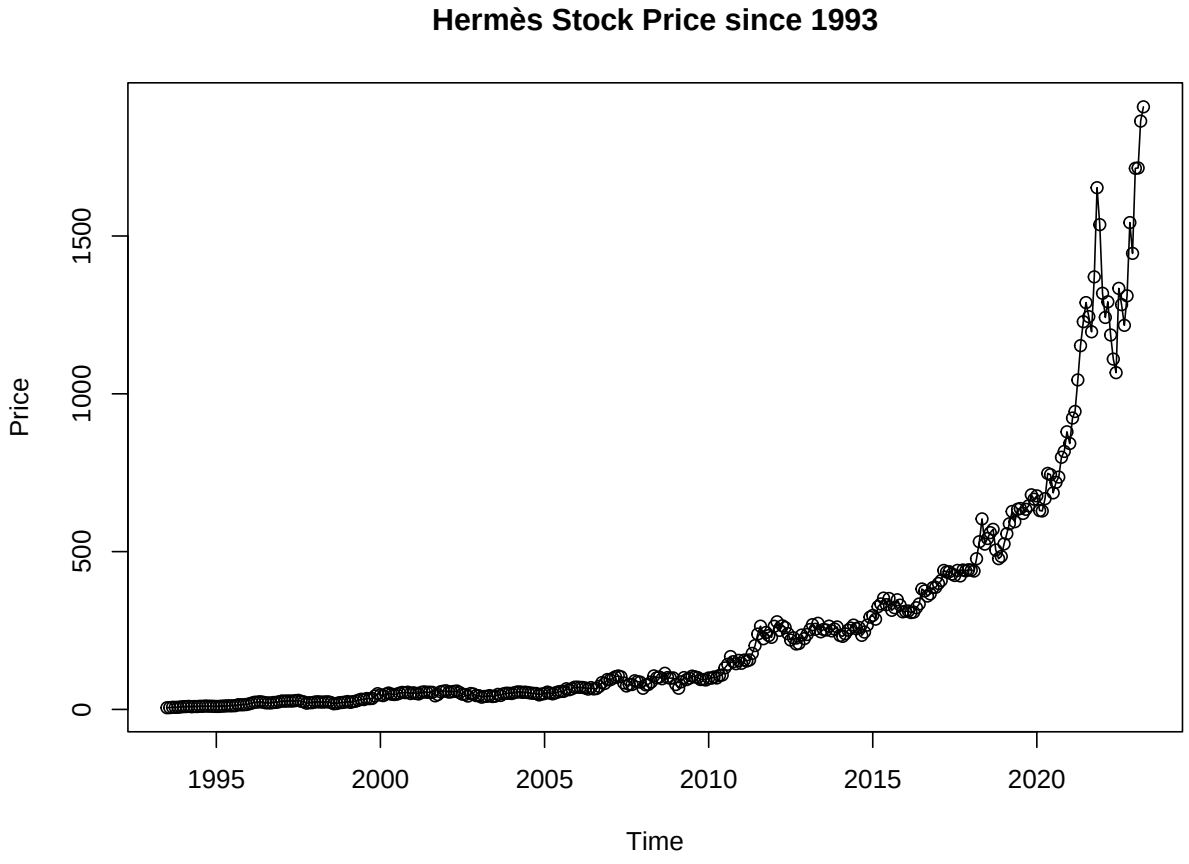

Figure 1: Hermès Stock Prices since 1993

Time Series Analysis

Based on the graph provided, we can conclude that the time series is non-stationary, characterized by a rising trend but no noticeable seasonality.

Firstly, we start by transforming the time series. We want to transform our data into a stationary time series (a time series with mean and variance independent of the time), thus:

```
# We create a time series object
Price_ts <- ts(Price_new)

plot(Date_new, Price_ts, type = 'l',
     xlab = "Time", ylab = "Price", main = "Hermès Stock Price Time Series")
```

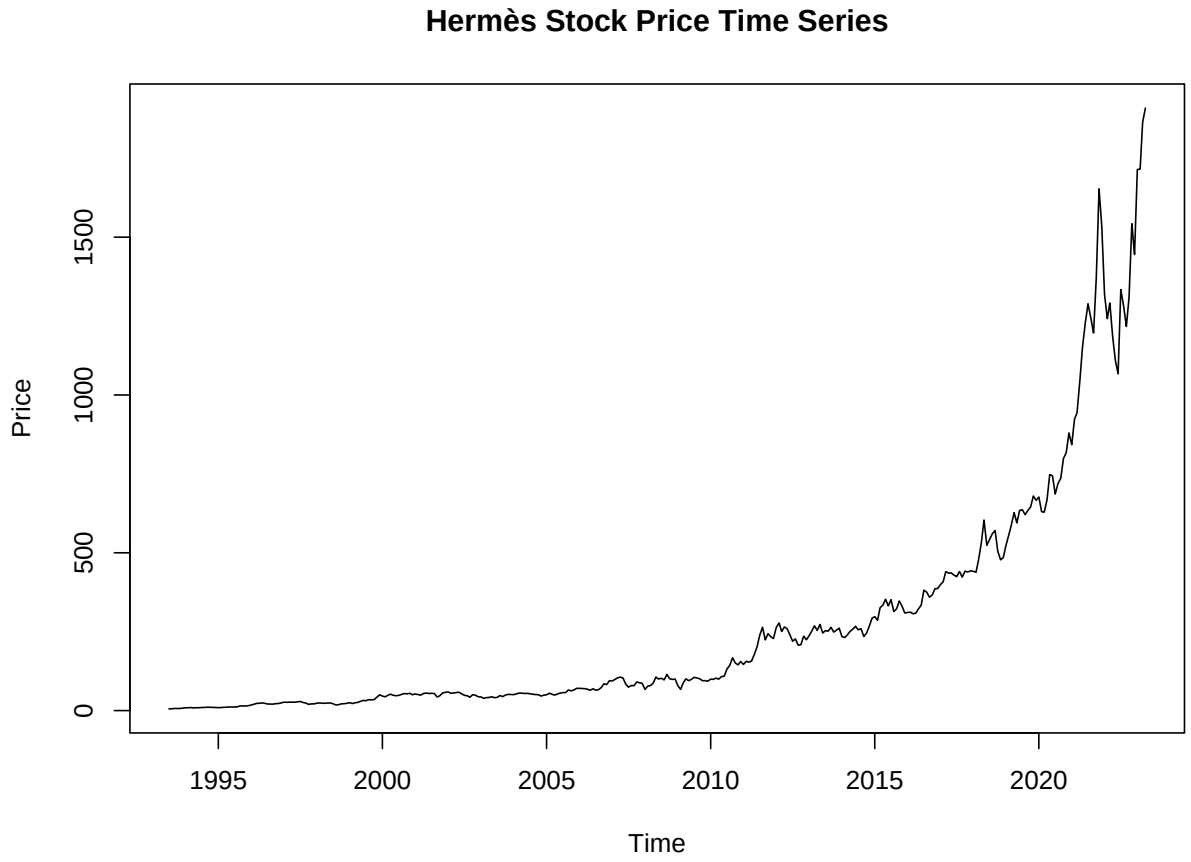

Figure 2: Hermès Stock Price Time Series

From the Time Series plot (fig. 1), we decide that we are dealing with an ARIMA(p, d, q) process. Our goal is to choose the best numerical values of p, d, and q.

The next step is taking the logarithm of the time series. This is a common data transformation technique used in time series analysis.

In our data set, we notice an increase in the variance overtime, which makes it difficult to forecast the series accurately. Taking the logarithm of the series can help to stabilize the variance, by compressing the range of values for large observations and expanding the range for small observations.

```r
# We take the log of the time series

Price_ts_log <- log(Price_ts)
plot(Date_new, log(Price_ts), type = 'l',
     xlab = "Time", ylab = "Price", main = "Hermès Stock Price Logged Time Series")
```

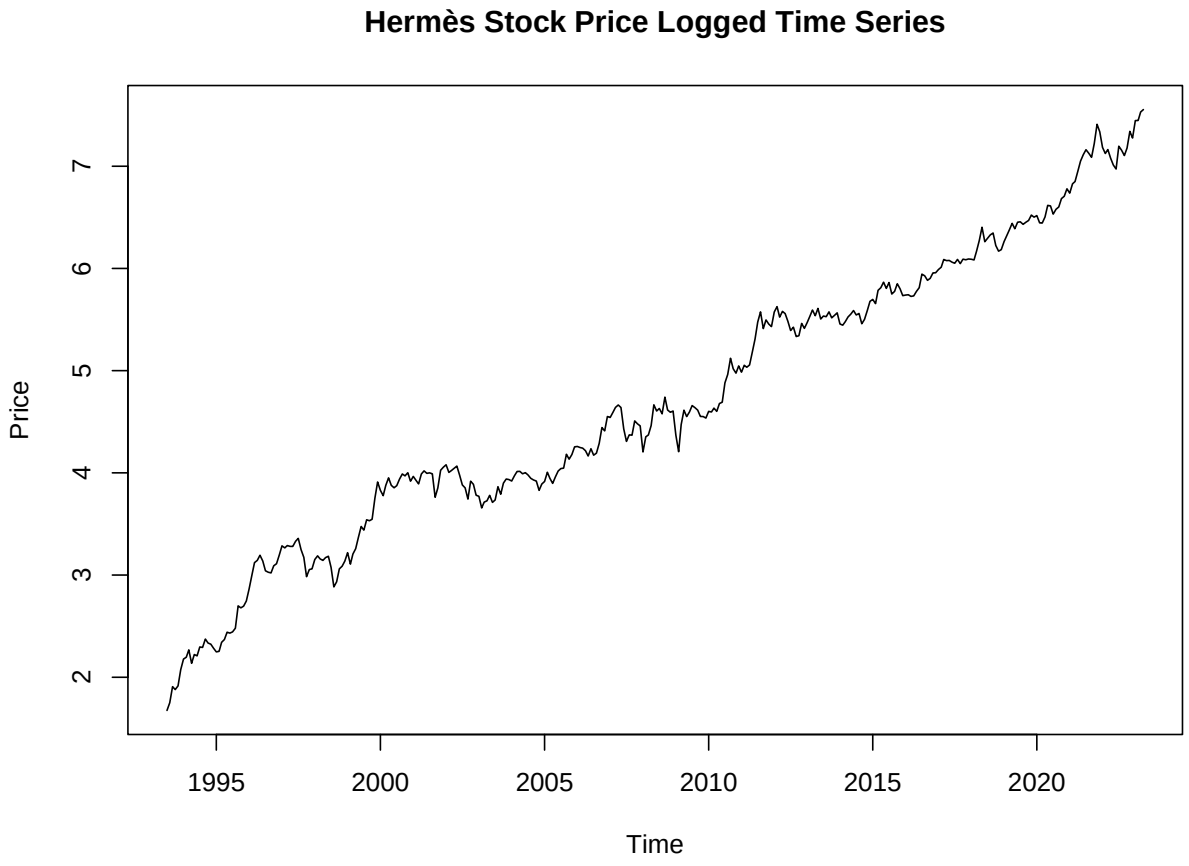

Figure 3: Hermès Stock Price Logged Time Series

Transformations such as logarithms can help to stabilize the variance of a time series. But when it comes to the mean of time series, differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

The differenced series is the change between consecutive observations in the original series, and can be written as

$$X'_t = X_{t+1} - X_t$$

where X_t is a time series.

At times, the differenced data might not appear to be stationary, and it might be required to perform a second differencing to achieve a stationary series:

$$X''_t = X'_{t+1} - X'_t$$

```r
# We take the first difference of the logged time series
diff1 <- diff(log(Price_ts))
```

The next step would be checking whether the first-order differenced time series is stationary or not. To do so, we can use the Augmented Dickey-Fuller (ADF) test.

The null hypothesis of the Augmented Dickey-Fuller test is that the time series is non-stationary. A small test statistic value suggests that the null hypothesis of a non-stationary time series cannot be rejected.

```r
library(tseries)
```

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

```r
# Perform the ADF test

test_1 <- adf.test(diff1)

# Print results

print(test_1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff1
## Dickey-Fuller = -7.5577, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
cat("ADF Statistic: ", test_1$statistic, "\n")
```

```
## ADF Statistic:  -7.557655
```

```r
cat("p-value: ", test_1$p.value, "\n")
```

```
## p-value:  0.01
```

```r
# We take the second-order difference of the logged time series

diff2 <- diff(diff1)

test_2 <- adf.test(diff2)

# Print results

print(test_2)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  diff2
## Dickey-Fuller = -11.085, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
cat("ADF Statistic: ", test_2$statistic, "\n")
```

```
## ADF Statistic:  -11.08481
```

```
cat("p-value: ", test_2$p.value, "\n")
```

```
## p-value:  0.01
```

By comparing the two ADF tests, we notice that both tests suggest that the time series is stationary since the p-value is $0.01 < 0.05$. But comparing the test statistics, we notice that the second test statistic is smaller than the first test statistic.

When the test statistic is very small, it means that the time series is closer to being stationary.

The first-order differencing removes most of the trend and seasonality of a time series. Therefore, we can conclude that the first-order difference time series is stationary, but still may have some degree of trend, that is not fully removed.

This is why we take the second-order differencing, which gives us a better stationary time series.

```
par(mfrow = c(1, 2))

plot(diff(log(Price_ts)), type = 'l',
     xlab = "time", ylab = "diff1", main = "Differenced Time Series ")
plot(diff(diff2), type = 'l',
     xlab = "time", ylab = "diff2", main = "Second-order differenced Time Series")
```

This conveys that the above is a second-order differenced model, hence $d = 2$, (see Fig.4).

The autocorrelation function (ACF) and partial autocorrelation function (PACF) plots are commonly used tools in time series analysis. The ACF plot shows the correlation between a time series and its lagged values at different lags, while the PACF plot shows the correlation between the time series and its lagged values after removing the effects of the intervening lags.

If the ACF plot decays exponentially, it suggests an auto-regressive $(AR(p))$ process.

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + Z_t$$

While, if the PACF plot decays exponentially, this suggests a moving average $(MA(q))$ process.

$$X_t = Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + \ldots + \theta_q Z_{t-q}$$

where Z_t represents a white noise $(0, \sigma^2)$.

```
par(mfrow = c(1,2))

# autocorrelation function
acf(diff2)

# partial autocorrelation function
pacf(diff2)
```

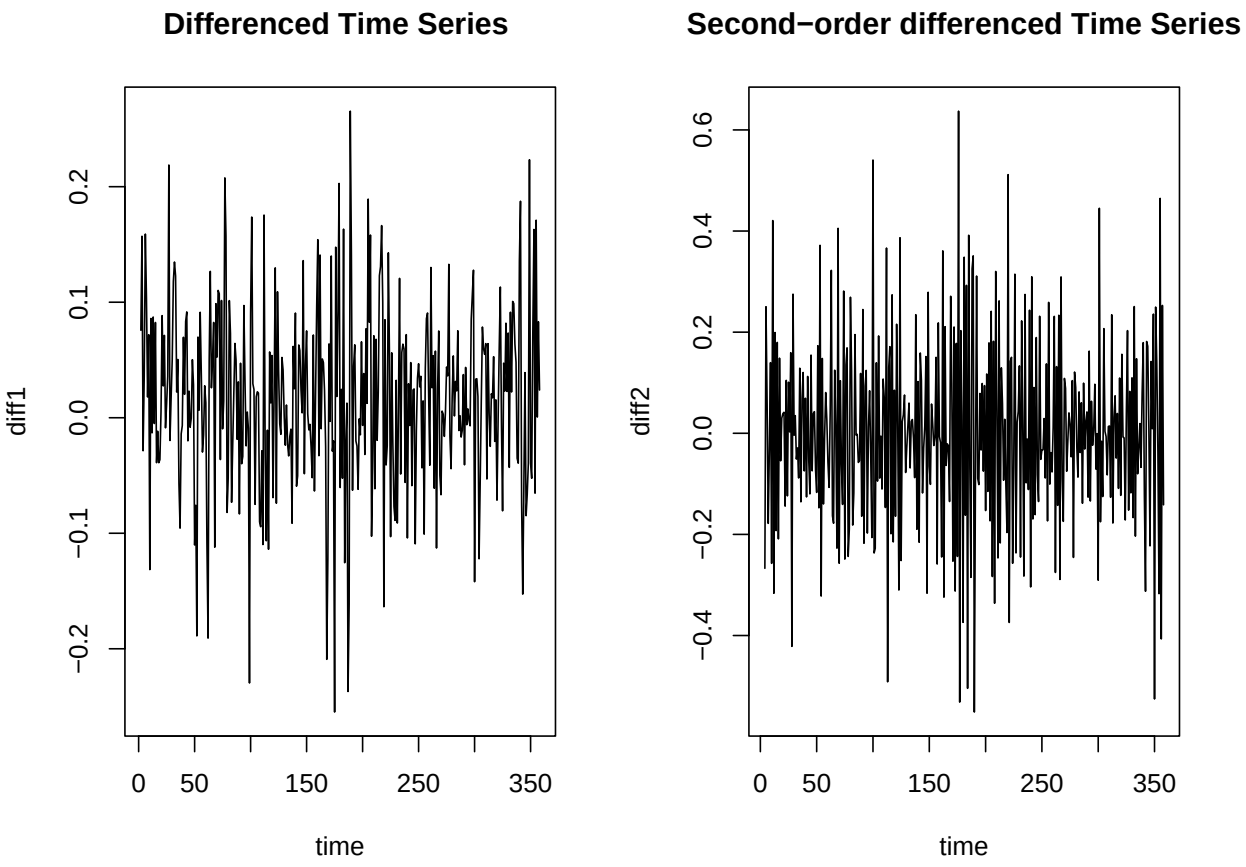

Figure 4: Differenced Time Series

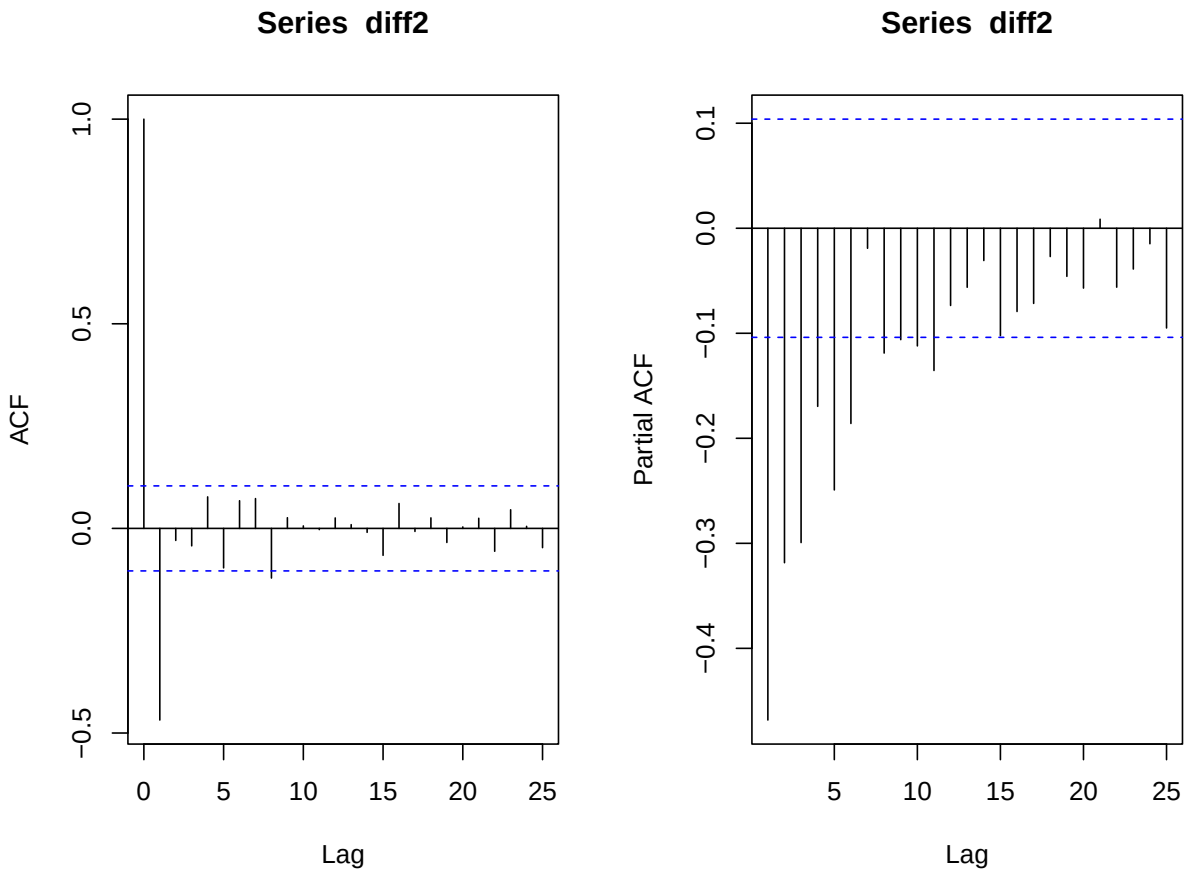

Figure 5: Autocorrelation and Partial Autocorrelation Functions

Based on the graphs of the ACF and PACF, (Fig.5), we conclude that the PACF is exponentially decaying. As a result, we are dealing with an MA process and $p = 0$. Since there is a significant spice at $lag1$ in the ACF, but none beyond $lag1$, this is an $MA(1)$ process

$$X_t = Z_t + \theta_1 Z_{t-1}$$

.

Once the process has been identified, we proceed to fit a model, using the "arima" function on R and the Maximum Likelihood Method.

```
# We fit an ARIMA (0, 2, 1) model
x.fit = arima(log(Price_ts), order = c(0, 2, 1), method ="ML")
print(x.fit)
```

```
##
## Call:
## arima(x = log(Price_ts), order = c(0, 2, 1), method = "ML")
##
## Coefficients:
##           ma1
##       -1.0000
## s.e.   0.0094
##
## sigma^2 estimated as 0.006274:  log likelihood = 394.6,  aic = -785.21
```

The ARIMA (0, 2, 1) model seems to be the best fit, considering that $AIC = -785.21$ is the smallest value. Additionally,

$$Z_t \sim (0, 0.006274)$$

, i.e normally distributed.

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```
# We generate forecasts for logged time series after 24 months

x.forecast <- forecast(x.fit, h = 24) # variable h represents the upcoming 24 months

plot( x.forecast, xlab = "time", ylab = "price",
      main = "Forecasts from Arima(0, 2, 1) for the upcoming 24 months")
```

```
# Then, we exponentiate the forecasted values to obtain forecasts for original time series

y.forecast <- exp(x.forecast$mean)

# We print 24 forecasted values for original time series
head(y.forecast, 24)
```

```
## Time Series:
## Start = 359
```

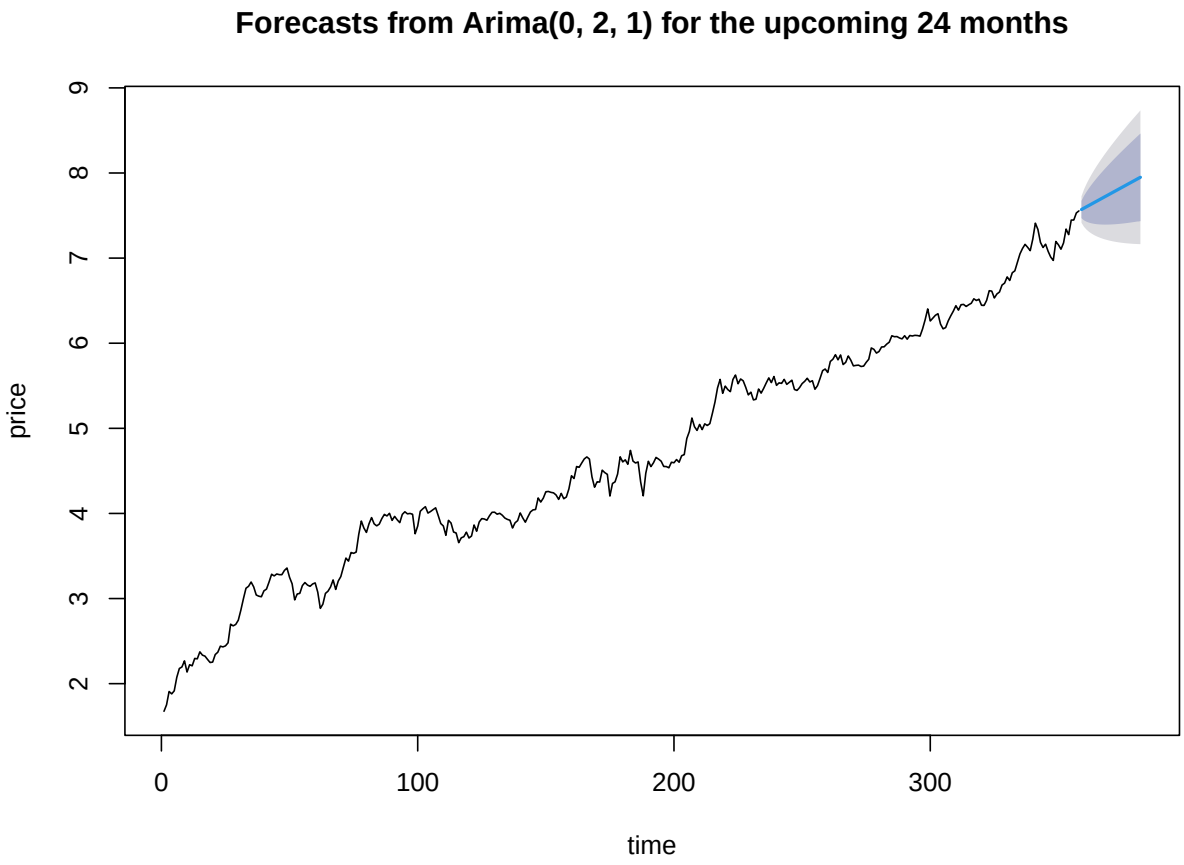

Figure 6: Forecast of ARIMA(0, 2, 1) for 24 months

```
## End = 382
## Frequency = 1
##  [1] 1940.902 1973.130 2005.893 2039.201 2073.061 2107.484 2142.478 2178.053
##  [9] 2214.219 2250.986 2288.363 2326.361 2364.989 2404.259 2444.182 2484.767
## [17] 2526.025 2567.970 2610.610 2653.959 2698.027 2742.827 2788.371 2834.671
```

Monte Carlo Simulation

The next step would be simulating the ARIMA model using Monte Carlo simulation.

```
# We simulate the distribution of future stock prices

mean_forecast <- mean(y.forecast) # the mean of the forecast model
mean_forecast
```

```
## [1] 2360.866
```

```
sd_forecast <- sd(y.forecast) # standard deviation of the forecast model
sd_forecast
```

```
## [1] 274.5604
```

```
sim_prices <- rnorm(1000, mean_forecast, sd_forecast) # this simulates 1000 stock prices
```

The next thing we want to do is calculate the expected returns for each simulated stock price.

For our first scenario, we choose the horizon, which is the length of time an investor plans to hold an investment, to be 2 years. Also, the risk-free rate, which is the rate of return an investor can expect to earn on a risk-free investment, we chose it to be 1. Thus, the investor is guaranteed to earn 1 of the return without taking any risks. The risk-free rate varies on several factors as economic conditions, duration of investment and inflation. Therefore, by choosing different values for the horizon and the risk-free rate, we can calculate different scenarios.

```
horizon <- 2 # investment horizon in years
rf_rate <- 0.01 # risk-free rate
# The expected return is the average return of the simulated
# stock price over the investment horizon of 2 years

# This is the return of the simulated stock prices
return <- (sim_prices - mean_forecast)/mean_forecast

# the compounded annual return
annual_return <- exp(mean(log(1 + return)))^horizon - 1 - rf_rate
```

After we defined the expected return of each simulated stock price, we repeat the same steps for $n = 1000$.

```
# We generate 1000 iterations
sim_iterations <- 1000
sim_results <- data.frame(matrix(NA, nrow = sim_iterations, ncol = 2))
colnames(sim_results) <- c("rf_rate")
```

```r
for (i in 1:sim_iterations) {
  sim_price <- rnorm(1, mean = mean_forecast, sd = sd_forecast)
  sim_return <- (sim_price - mean_forecast)/mean_forecast
  sim_annual_return <- exp(mean(log(1 + sim_return)))^horizon - 1 - rf_rate
  sim_results[i, 1] <- sim_annual_return
}
```

```r
# we plot distribution of investment returns
plot(density(sim_results$rf_rate),
     main = "Distribution of Expected Annual Return over 2 years with risk-free rate 1%",
     xlab = "Expected Annual Return")
```

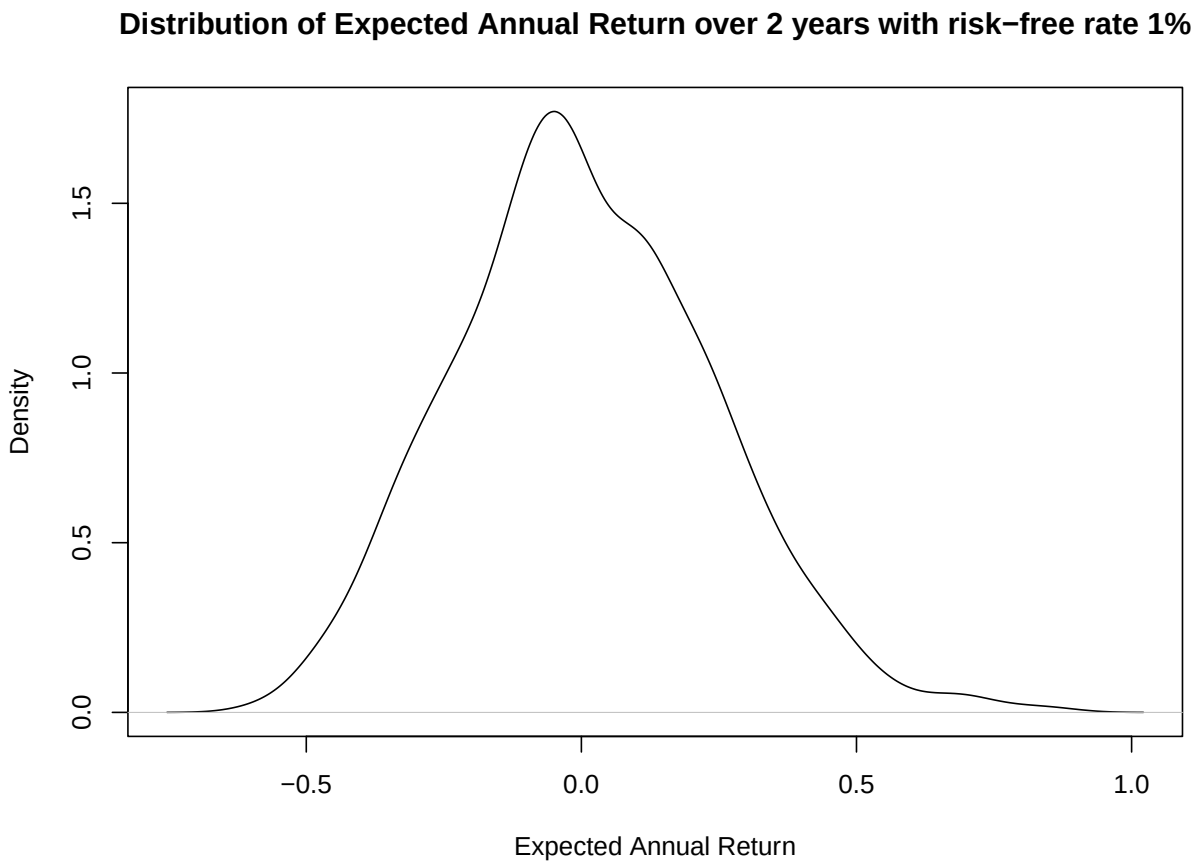

Figure 7: Expected Annual Return over 2 years with risk-free rate 1%

We can choose a different scenario. For instance, we can take a new risk-free rate to be 5 % instead. Then, we repeat the same process. (A comparison between different values of the risk-free rate is given below)

In conclusion, the distribution of expected annual returns informs us about the range of potential returns from an investment. We can learn about the risks and potential rewards of an investment by evaluating the shape and spread of the distribution.

The plot (see Fig.7) gives us an idea of the range of the potential return expected from the investment. A skewed distribution toward larger returns may imply a higher risk investment, whereas a more symmetric distribution may indicate a lower risk investment.

The curve (Fig.7) seems to be somewhat wide and flat, suggesting a high degree of uncertainty in the expected returns.

Furthermore, there are various other methods and measures that assist us to comprehend and assess better the risk and return of our investment.

One of them would be the Sharpe Ratio, which is a measure of risk-adjusted return and it measures the excess return of an investment over the risk-free rate relative to its volatility.

The Sharpe Ratio is given by the formula

$$Sharpe = \frac{R_p - R_f}{\sigma_p}$$

, where R_p is the return of portofolio, R_f is the risk-free rate and σ_p is the standard deviation of the portofolio's return.

```
# Define the terms

rf_rate <- 0.01 # risk-free rate
R <- mean(sim_results$rf_rate) # the average annual return of the simulated stock prices
S_D <- sd(sim_results$rf_rate) * sqrt(2) # the standard deviation

Sharpe_Ratio <- (R - rf_rate) / S_D
Sharpe_Ratio
```

```
## [1] -0.01216242
```

In the case when the risk-free rate is taken to be 1%, the Sharpe Ratio is -0.3178729. A negative Sharpe Ratio indicates that on average, the investment's return might be lower than the risk-free rate. A Sharpe ratio of -0.3178729 indicates that the simulated stock prices are expected to generate a return that is 0.3178729 standard deviations below the risk-free rate over a 2-year investment horizon.

This shows that the investment may not be a suitable decision when weighing the expected return versus risk, as the projected return is insufficient to compensate for the higher risk involved.

Another method for calculating the risk and loss of an investment would be Value at risk (VaR). The value at risk (VaR) statistic assesses the extent of potential financial losses inside a firm, portfolio, or position over a given time period.

VaR modeling determines the entity's potential for loss as well as the likelihood that the defined loss will occur. VaR is calculated by evaluating the amount of potential loss, the probability of the amount of loss occurring, and the time frame.

Returning to the scenario we were studying previously, we want to calculate the 95% VaR for a 2-years investment horizon.

```
horizon <- 2 # investment horizon in years
rf_rate <- 0.01 # risk-free rate
initial_investment <- 100000 # initial investment

# Calculate the investment return for each simulated stock price
returns <- (sim_prices - mean_forecast)/mean_forecast

# Sort the returns
sorted_returns <- sort(returns)
```

```r
# Determine the percentile of the sorted returns
percentile <- qnorm(0.05, mean = mean(returns), sd = sd(returns))

selected_return <- sorted_returns[which(sorted_returns <= percentile)
                                  [length(which(sorted_returns <= percentile))]]

# Define VaR
var <- abs(initial_investment * selected_return)

cat("The VaR for a two-years investment horizon and a 95% confidence level is:", var, "\n")
```

```
## The VaR for a two-years investment horizon and a 95% confidence level is: 19490.06
```

The VaR (value at risk) of 18655.95 implies a 5% chance of the investment losing at least that amount over the specified investment horizon (in this case, two years) with 95% confidence. For example, if you invest $\$100,000$ for two years, there is a 5% chance that the investment will lose at least $ 18655.95.

In conclusion, several elements are considered when measuring the risk of an investment, such as corporate performance, industry trends (in our case, fashion trends), market circumstances, regulatory environment, volatility, and so on.

The past performance of the stock price, market trends, and volatility, as indicated in the ARIMA model, are taken into account to estimate the risk of investing in forecasting Hermès Stock Prices. Furthermore, as a measure of the overall risk of investing in the company, the risk-free rate of return is taken into account when calculating the expected return and compounded annual return. The chart provided below (Fig.8) illustrates the varying impact that different risk-free rates have on each distribution.

To summarize, an investment with a wide normal curve, a negative Sharpe ratio and a positive VaR suggests an investment that is risky and most likely does not generate a positive return that compensates for the taken risk.

To summarize, an investment with a wide normal curve, a negative Sharpe ratio and a positive VaR suggests an investment that is risky and most likely does not generate a positive return that compensates for the taken risk.

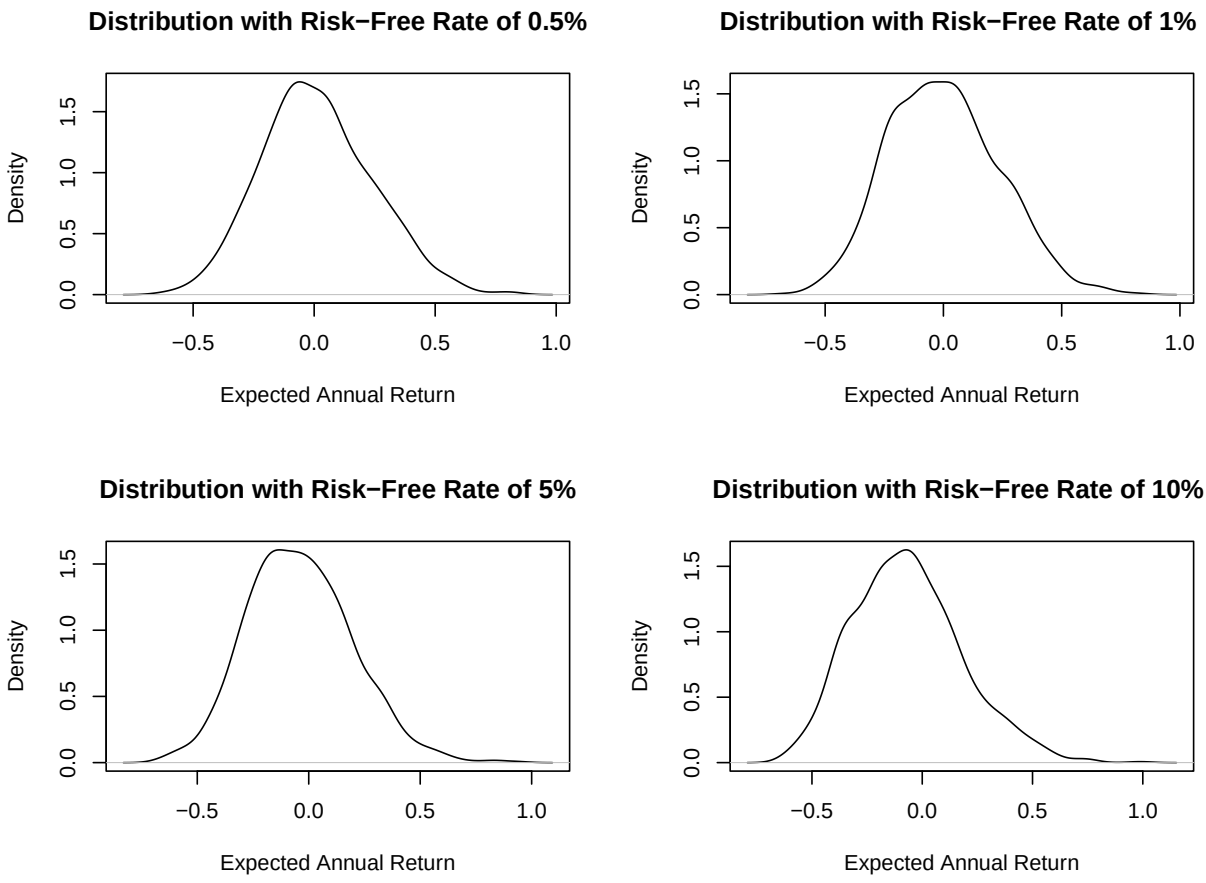

Figure 8: A Comparison of Expected Annual Returns over 2 years with Different Risk-Free Rates

References

1. Rizzo, Maria L. (2019). Statistical Computing with R, Second Edition
2. Brockwell, Peter J., Davis, Richard A. (2016). Introduction to Time Series and Forecasting, Third Edition
3. Prabhakaran, S. (2021). ARIMA Model - Complete Guide to Time Series Forecasting in Python (https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)
4. CFI Team. (2023). Annual Return (https://corporatefinanceinstitute.com/resources/capital-markets/annual-return/)
5. Stationarity Testing (https://rpubs.com/richkt/269797)
6. Fernando, J. (2022). Sharpe Ratio Formula and Definition With Examples (https://www.investopedia.com/terms/s/sharperatio.asp)
7. Kenton, W. (2023). Understanding Value at Risk (VaR) and How It Is Compted. (https://www.investopedia.com/terms/v/var.asp)