S & P 500 index performance: Combining ARIMA + NNAR and GARCH Techniques

Andy Chen

March 2025

Abstract

This study investigates the return dynamics and volatility of the Standard and Poor's 500 (S & P 500) stock market index in the period 2000-2015. We use the ARIMA, NNAA, and GARCH model to forecast the return and volatility of the index for the next 12 months. Understanding these fluctuations is crucial in investing, decision-making, and risk management.

In the process of forecasting returns, we applied the ARIMA model in the beginning, which captures linear dependencies in time series data. However, its assumption of homoscedasticity limits its ability to model volatility clustering. Hence, we integrate a Neural Network Regressive Model(NNAR) to enhance predictive accuracy, which is more intelligent in capturing non-linear relationships within the data and improves short-term forecasting. In addition, we employed the GARCH model to capture time-varying volatility. Although GARCH(1,1) model can be a good fit for modelling volatility and NNAR improves short-term return forecasts, it is still difficult to predict a long-term S & P 500 index in both expected return and volatility due to the complex and stochastic nature of financial markets.

1 Introduction

In the global economy, the stock market plays a vital role. People's investment in the stock market gives the market vitality. In order to meet the investor's expectations, Accurate stock price forecasting is essential for informed investment decisions, risk management strategies, and market regulations. This project focuses on the Standard and Poor's 500 (S & P 500) stock market index, which is a stock market index tracking the stock performance of 500 of the largest companies listed on stock exchanges in the United States. It is the well-known

and popular equity indices, accounting for many mega-companies that we are familiar with such as Apple, Microsoft, Amazon, etc. Thus, given the great influence of these companies and the rapidly growing industry, accurately forecasting their stock index movements is of great interest to market participants. Despite these challenges, this study demonstrates the strengths and limitations of statistical and machine learning models in stock market forecasting and highlights the need for more advanced hybrid approaches.

This project examines time series forecasting methods to forecast the future S & P 500 stock index. We evaluate the performance of ARIMA (Autoregressive Integrated Moving Average), GARCH (Generalized Autoregressive Conditional Heteroskedasticity), and a neural network (NNAR) model for stock index forecasting. ARIMA is widely used for modeling time-series data, while GARCH effectively captures volatility clustering, a key characteristic of financial markets. However, both models have limitations in handling nonlinear patterns. To address this, we implement an NNAR model based on the ARIMA model, which leverages neural networks to learn complex relationships within the data. By integrating these approaches, we aim to develop a more robust forecasting framework for S & P 500 index movements.

2 Data

The dataset covers the stock index of Standard and Poor's 500 (S & P 500) from January 1, 2000 to Dec 1, 2015, consisting of a period of 15 years. To compose the data easily, we selected the monthly data that is on the first day of each month. The dataset contains the real and nominal index, but we will mainly focus on the real one, which reflects inflation-adjusted values, providing a more accurate measure of the index's true growth. This allows us to compare the index's performance over different periods without the distortions caused by inflation.

As I have aforementioned, S & P 500 stock market index tracking the stock performance of 500 of the largest companies listed on stock exchanges in the United States. The variation from the real index will influence the expectation and confidence of the U.S economy. As we know, there is a global financial crisis that began in the United States in 2008. Thus, analysis of S & P 500 stock index in the period I chose provides valuable insight into historical events, then we can forecast similar patterns to have an early warning of the foreseeable disaster.

The dataset is coming from the MacroTrends(website link: MacroTrends).

It represents the historical S & P 500 index information over the past 90 years. The data provider collects the data from the S & P and Dow Jones Indices LLC and retrieves this information in order to make it available to users through their platforms.

3 Methodology

3.1 ARIMA Model

The ARIMA is a widely used model in time series forecasting. It consists with AR(autoregression), I(Integrated) for differencing, and MA(moving average). The principle of ARIMA is based on the analysis of the relationship between an observation and a number of lagged observations and errors. Since the S & P 500 dataset contains trends and non-seasonal patterns, ARIMA is a suitable model to use. The equation of ARIMA(p, d, q) is defined by:

$$\sum_{i=0}^{p} B^i(1-B)^d x_t = \delta + \sum_{j=0}^{q} \theta_j B^j w_t \quad t \in Z \tag{1}$$

Among them, B is the backshift operator, (1-B) is the difference operator, d is the level of differentiation. In our case wo only need to set d =0 because during the model fitting, we are going to do the stationary check so it is unnecessary to do differencing and ARIMA(p, 0, q) is way simpler to interpret than within value of d.

3.2 SARIMA Model

SARIMA model is similar to ARIMA, but the S is denoted as the seasonal effect. It is composed of (p,d,q) $\times$ (P, D, Q). The P is a seasonal order determined by the seasonal AR, and P is a seasonal order determined by the seasonal MA. If there is a seasonal differencing part, we will also consider the D when necessary. The SARIMA(p, d, q) $\times$ (P, D, Q) is defined by:

$$\Phi_P(B^s)\phi(B)\nabla_s^D\nabla^d x_t = \delta + \Theta_Q(B^s)\theta(B)w_t \quad t \in Z \tag{2}$$

where s is the seasonal period.

3.3 NNAR model based on ARIMA

Neural networks have gained popularity in time series forecasting due to their ability to capture complex, nonlinear relationships in data. In this study, we implement a Neural Network AutoRegressive (NNAR) model with the character of feedforward(FNN). This model can be trained to approximate the underlying patterns in data, and has more flexibility and accuracy since it uses the basic concept of machine learning to deal with data.

3.3.1 FNN

In the feedforward neural network, there are 3 components. One is the input layer, it receives p-lagged observations from the time series as input features. Then, the data received will be processed in the hidden layer, which is composed of one or more neurons with activation functions that can introduce nonlinearity to the model. After this procedure, the predicted value will be produced through the output layer at the time t. Figure 1 below shows the structure of the FNN, and I will discuss further in the section below.

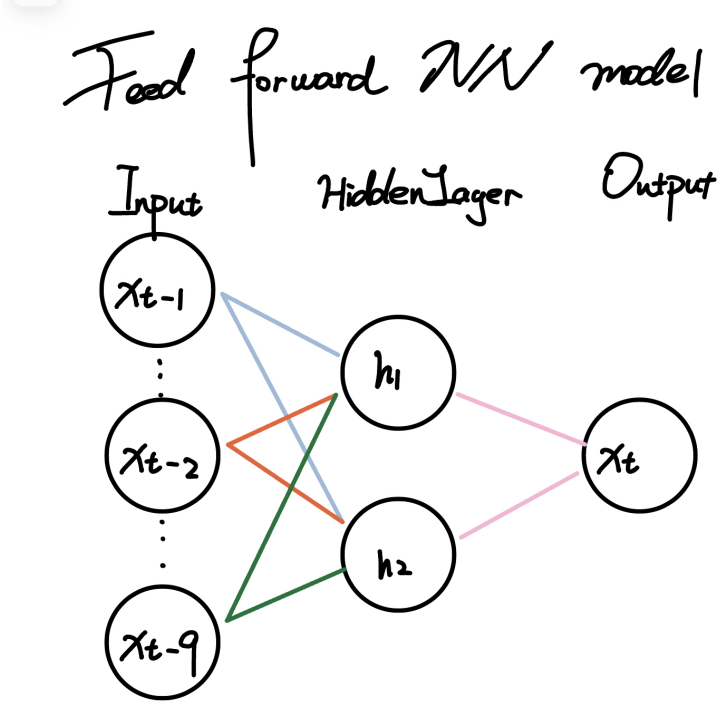


Figure 1: The prediction by using NNAR model

3.3.2 ARIMA and FNN

We define the FNN function by:

$$\hat{x_t} = \sum_{j=1}^{q} w_{1,j} x_{t-j} + \sum_{i=1}^{p} w_{2,i} h_i \quad t \in Z \tag{3}$$

Among them, q is the number of inputs, p is the number of neurons, h_i is the activation output of neuron i, and h_i have a function f_i defined as $h_i = f_i(\sum_{j=1}^{q} w_j^{(1)} x_{t-j})$ which is the processing function in the hidden layer as the Figure 1 shown above.

Now recall the ARIMA model function from 3.1:

$$\sum_{i=0}^{p} B^i (1-B)^d x_t = \delta + \sum_{j=0}^{q} \theta_j B^j w_t \quad t \in Z \tag{4}$$

As we can see, the ARIMA process and NNAR FNN process have some similarities. In the FNN model, the first term $w_{1,j}$ is the input of past number of input q with x_{t-j}, and ARIMA is the past number of input p x_{t-i} with the procedure of backshift. We use the same data for these two processes, so the number of input p, q are the same. The only difference is that, FNN is more intelligent in the procedure of forecasting. So it does not have the manipulation of differentiation and the White noise term, and is replaced by the second term of the hidden layer h_i.

3.4 GARCH Model

GARCH model is a time series model that is mainly used for forecasting and modeling the volatility of financial returns and other time series data characterized by changing variances over time. This is also called heteroskedasticity by definition. The GARCH model is an extension of ARCH model, by further ensuring the variance from the past to interfere with the future variances. By this step, the GARCH model will have better performance in capturing volatility clustering commonly observed in financial markets. Within that, the result of high volatility tends to be caused by high volatility, and so as low.

The GARCH(p,q) model consists of the number of lagged conditional variance(p) and the number of lagged squared residuals(q). It is defined as:

$$a_t = \sigma_t \varepsilon_t \tag{5}$$

where σ_t is:

$$\sigma_t = \sqrt{\omega + \sum_{i=1}^{p} \alpha_i a_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2} \tag{6}$$

Generally, for the time series model, we let a_t be GARCH(p,q) and we use it as the noise term in an ARIMA(p, d, q) model. This is also an ARIMA(p, d, q) + GARCH model analysis process.

4 Results

In the analysis process, we will select S & P 500's previous index as the dataset and try to forecast the new index in the next 12 months. In general, the stock index will not perform any seasonality, but we still need to check the pattern in the plot to determine whether to use SARIMA or ARIMA model. And then we will try to use the GARCH model to improve our prediction.

4.1 Building the ARIMA Model

The index we pickup is the Standard and Poor's 500 stock market index (S & P 500). We check the Box-Jenkins approach to build ARIMA / SARIMA model first.

4.1.1 Time series Data Plot

We first plot the time series data for the S & P 500 index. Usually the index's daily movements can be visualized by the candlestick or line charts. We chose the dataset from 2000 to 2015, which is a 15-year period. So the everyday's data point is a vast number to perform. To address this, we select the **real** index value on the first day of each month as a reference, which helps reduce the number of data points while clearly presenting long-term trends

The time series plot for the S & P 500 index, using the real index values, is shown in Figure 2.

From the plot we can see that the distribution of the graph is highly randomized, which conveys that the time series of S & P 500 might not be stationary. Then we need to do the ADF test and the QQ plot to have further clarification. The Figure 3 above showing the QQ plot of S & P 500 index original data. We can find that the simulated data has a heavy tail that deviated from the fitted line, which suggests the data we are checking is not normal and we need to have a transformation to alleviate the influence from the outliers. For the ADF

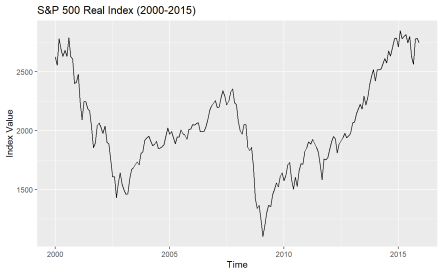


Figure 2: Time Series plot for S & P 500 index

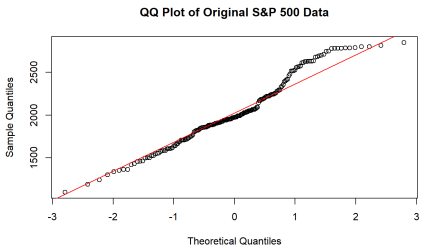


Figure 3: QQ-plot for S & P 500 index

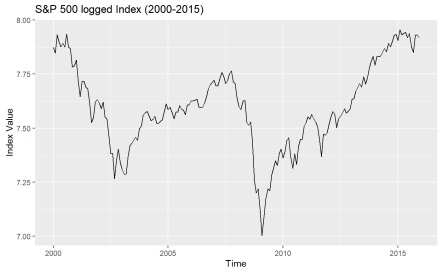


Figure 4: Time Series plot for logged S & P 500 index

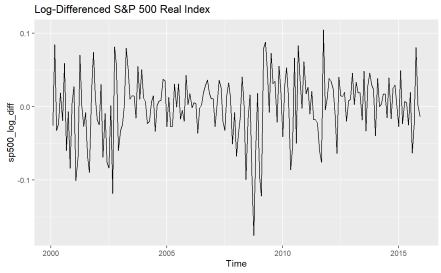


Figure 5: Time Series plot for log differentiated S & P 500 index

test, we set the null hypothesis that the time series is non-stationary versus the alternative hypothesis that the time series is stationary. In our test, the p value is 0.4456, which means the time series is nonstationary and we fail to reject the null hypothesis.

4.1.2 Transformation

From the result that QQ plot and ADF test reveal in section 4.1.1, we know that the time series of S & P 500 is not stationary. Therefore, we have to do transformation to make the time series stationary before we apply the ARIMA model. In our case we applied the log transformation. The plot is shown in Figure 4. The graph is similar to Figure 2, where implies we need to do a differentiation. Thus, after a 1 level differentiation, the new graph is shown in Figure 5, which is a log-differentiated transformed S & P 500 index. Comparing to the previous two data, it becomes stationary. The ADF test of the log differentiated data shows the p value is less than 0.01, which proves the data after the transformation is stationary.

4.1.3 Estimation of Parameter and Prediction

Since the log-transformed data becomes stationary, we can set the difference level d = 0. Now we need to check the ACF and PACF plot to determine the p and q in the ARIMA model.

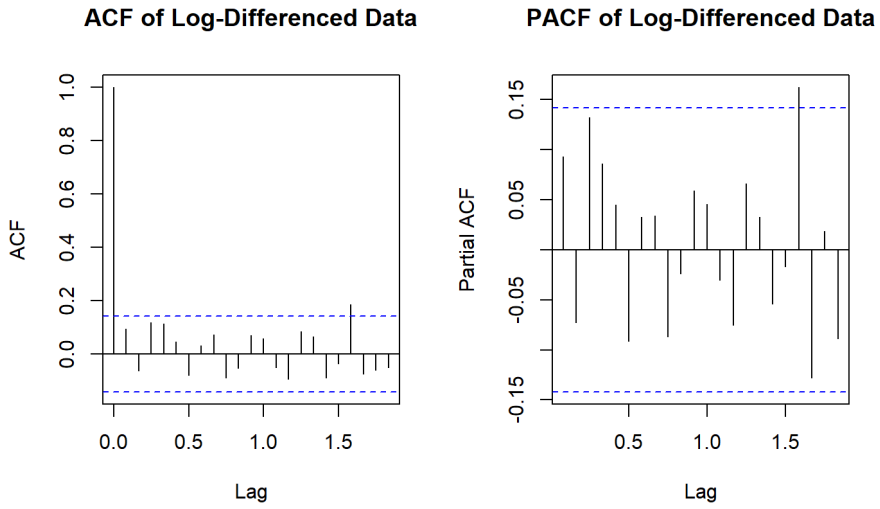


Figure 6: The ACF and PACF plot for log transformed S & P 500 index

As the plot showing above(Figure 6), the lags in the ACF plot is lying in the interval with no extreme values.So does the PACF in this case. Since there's no clear cut off or decay in ACF and PACF, it suggests the S & P 500 index data do not have seasonality. Therefore the AR(p) and MA(q) can only be 0 as p and q = 0. The result that came out from the 'auto.arima()' function is the same as the determination, which is ARIMA(0,0,0). The summary of ARIMA(0,0,0) of S & P 500 index data is:

```
    Series: sp500_log_diff
ARIMA(0,0,0) with zero mean
sigma^2 = 0.001935:  log likelihood = 325.62
AIC=-649.24   AICc=-649.22   BIC=-645.99

Training set error measures:
    ME   0.0002353081     RMSE  0.0439916

    MAE 0.03341372        MPE 100  MAPE   100
```

```
MASE 0.7298417          ACF1  0.0929069
```

From the ARIMA summary, the $Sigma^2$ value is very small(0.001935), indicating that the level of residual variance is very low. The AIC and BIC are both very negative. The three values the ARIMA model conveys that the model fit the S & P 500 data well. However, though AIC and BIC are low, the ARIMA(0,0,0) model does not capture any meaningful structure in the data — it only models white noise. This result shows that the forecast of the data will be a straight line(Figure 7):

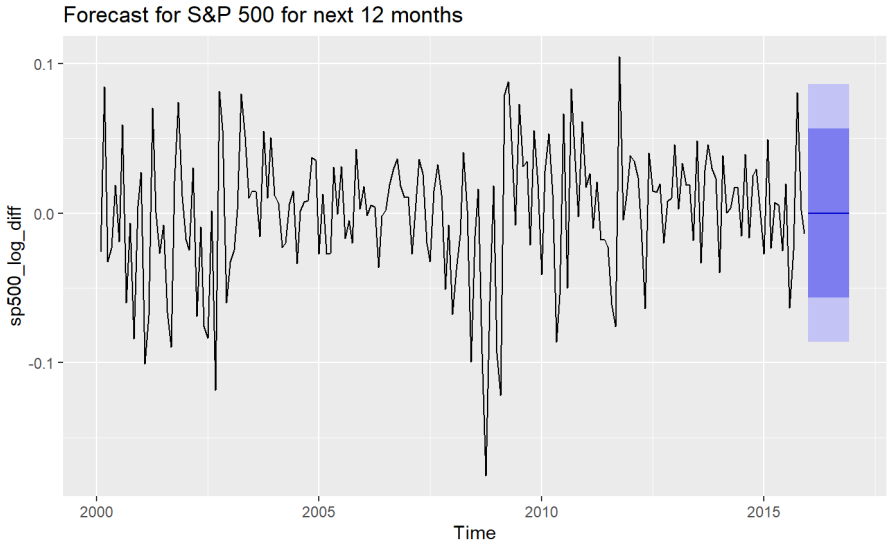


Figure 7: The prediction by using ARIMA(0,0,0) model

The model is only driven by white noise, which is impossible in real life. So we need to use the GARCH model in order to improve our performance of forecasting the S & P 500 index data.

4.2 ARIMA(0,0,0)+GARCH(p,q) Model

4.2.1 GARCH Model set up

In the ARIMA model, we focus on capturing the mean of the time series. This has an assumption that our stock index will only follow a constant variance. But in the real life stock market situation, the index is unstable and changes over time. Therefore, it is unrealistic that the forecasting of the stock index is

just a straight line. To figure out this, will apply the GARCH(p_V, q_V) as the noise term in an ARIMA($p + M$, d, q_M) model. We will call such a process an ARIMA($p + M$, d, q_M) + GARCH(p_V, q_V) model. [1].

In the beginning of the GARCH model set up, we will assume the distribution follows the Gaussian Distribution and start with the value p_V and q_V equal to 1. The result is processed by using the 'rugarch()' package. [2] In the result, we mainly focus on the Adjusted Pearson Goodness-of-fit Test. This test is useful to check if the error term follows the normal distribution. Typically, if the p-value is less than 0.05, we reject the null hypothesis and conclude that the model does not fit the data well. We selected the result from the GARCH test below:

```
Adjusted Pearson Goodness-of-Fit Test:
        group statistic p-value(g-1)
  1    20     26.63       0.1135
  2    30     29.16       0.4569
  3    40     44.11       0.2645
  4    50     54.09       0.2862
```

In our case, since all the p values are greater than 0.05, we conclude that there is no strong evidence to reject the null hypothesis. Thus, the model GARCH(1,1) appears to be a good fit for the S & P 500 index data. Now, we can use this model to predict the time series.

4.2.2 prediction

Since the GARCH model has a good fit, we can use the ARIMA(0,0,0)+GARCH (1,1) to predict the S & P 500 Stock index for the next 12 months. The result is shown in the Figure 8 below:

From the prediction plot, we can find several things very noticeable:

1. In the first month, the index will increase relative to the previous month, but the increase will be small.

2. After the first month, we expect remain at the same level for the remaining months.

Comparing the prediction to the first one, it has a numeric value revealed. This is an improvement compared to the straight line in the ARIMA(0,0,0)

[1] V is denoted as the variance and M is the mean
[2] The full GARCH test result is in the Appendix page.

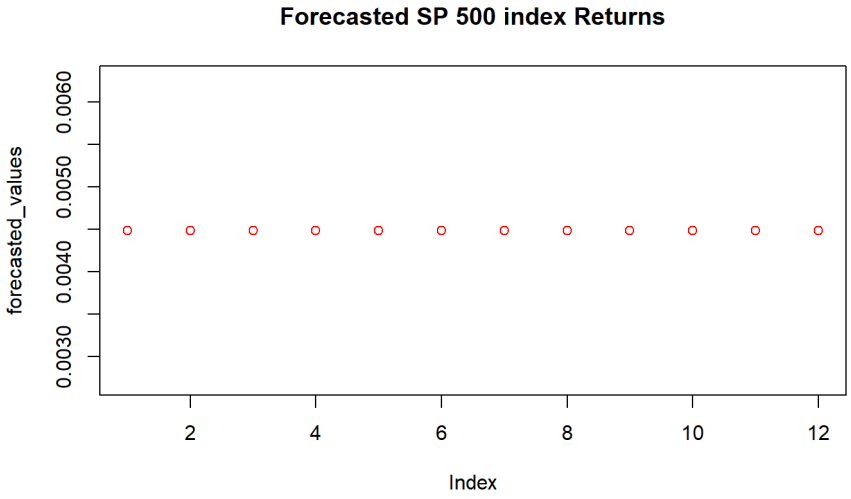


Figure 8: The prediction by using ARIMA(0,0,0) + GARCH(1,1) model

model. However, this model does not really meet our expectation. So we can enhance the situation through the NARR model.

4.3 NARR model

Since the ARIMA(0,0,0) + GARCH(1,1) does not met our expection, we can enhance our model by capturing some non-linear patterns through the NARR model. We used 'nnetar()' function in the forecasting package to let the model auto select the hidden layer to process with. And based on the result given, the system selected a NNAR(3,2) process, with a 3 time step and 2 hidden layers in it. we can have a new graph of the forecast S& P 500 index data, as shown in Figure 9.

As we can see, the forecast index graph from the NNAR model is better than the previous two with a fluctuation. But the variation forecast value is still minor and became a straight line in the long term period, so we need to find out why this happens.

One reason is that, the difficulty of predict the future stock index is very high. Because the forecast is 12 months, the value can be varied easily. Also, we only selected the 1st day of each month, the result can not be precise to that level.

Another reason is that, the ARIMA model is not an effective model to predict

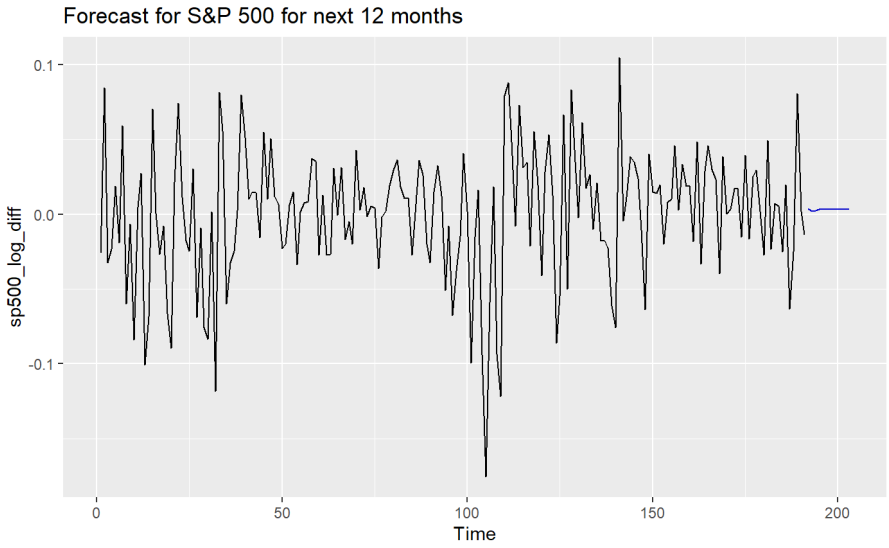


Figure 9: The prediction by using NNAR model

the stock index. As we have mentioned in the previous section, we have an assumption of ARIMA(p, d, q) that our stock index follows a constant variance, which is contradict to the real life. Even though we improve the model by adding the GARCH(p_V, q_V), due to the ARIMA model's p_M, q_M remains 0, the combined model is still driven by the white noise.

5 Conclusion of the study

In this study, we applied time series forecasting techniques to predict the future performance of the S & P 500 index. In the whole process we started with the ARIMA model, which is mainly used for time series forecasting. However, we discovered that the forecasting by the ARIMA model is a straight line in the assumption of constant variance, which is unrealistic in the real life.

To address this limitation, we let GARCH model engage in the forecasting, which accounts for time-varying volatility. The ARIMA(0,0,0) + GARCH(1,1) model provided a better fit to the data compared to the ARIMA model only. By modeling volatility, the combined model was able to reflect the market index more accurately, but only achieved a better performance in the short term. We finally tried the NNAR model with a 3 time step input and 2 hidden layers. The final result performed better than the other 2 time series models, with a

visible fluctuation compared to the ARIMA model only. However, this model also does not really meet our satisfaction for long-term forecasting due to the complex nature of financial markets. Hence the prediction of long-term stock indices is still challenging and we may need a more advanced model to figure out the difficulty.

For the further investigation, we could explore more advanced forecasting techniques, such as machine learning and deep learning models, to better capture the nonlinear relationships in stock price movements. Furthermore, we can also refer to some external information in the market, such as government policy, news sentiments, to enhance our accuracy. While there's a limitation in our research, the classic model like ARIMA and GARCH provides a stepping stone into more robust and adaptive modeling approaches for stock index prediction.

A Reference

S & P 500 Index - 90 Year Historical Chart. https://www.macrotrends.net/2324/sp-500-historical-chart-data. Accessed 11 Mar. 2025.

B R code

R code for S & P 500 data set

Andy Chen

2025-03-13

```
library(readxl)
library(ggplot2)
library(mvdalab)
library(astsa)
library(forecast)
```

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

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:astsa':
##
##     gas
```

```
library(tseries)
library(xts)
```

```
## Warning: package 'xts' was built under R version 4.4.3
```

```
## Loading required package: zoo
```

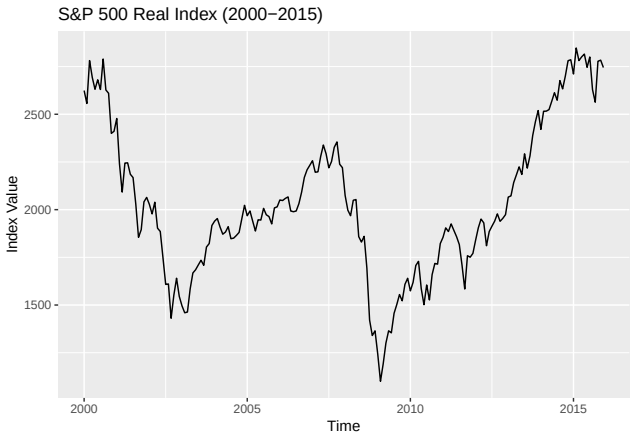
```
##
## Attaching package: 'zoo'
```

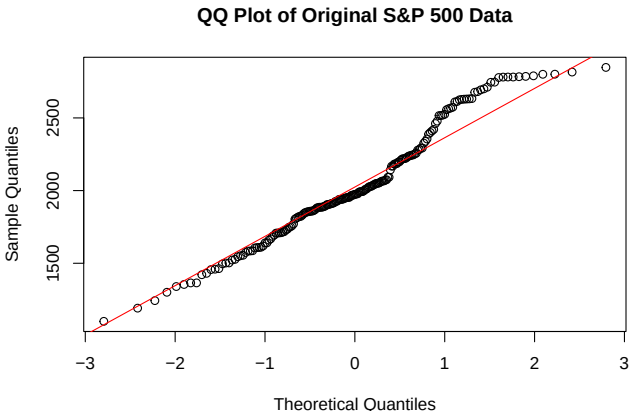
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
SP500 <- read_excel("S & P 500 2000 - 2015.xlsx")
## Step 1
SP500$date <- as.Date(SP500$date)
SP500_real <- ts(SP500$real, start=c(2000,1), frequency=12)

autoplot(SP500_real) + ggtitle("S&P 500 Real Index (2000-2015)") + ylab("Index Value")

qqnorm(SP500_real, main="QQ Plot of Original S&P 500 Data")
qqline(SP500_real, col="red")
```



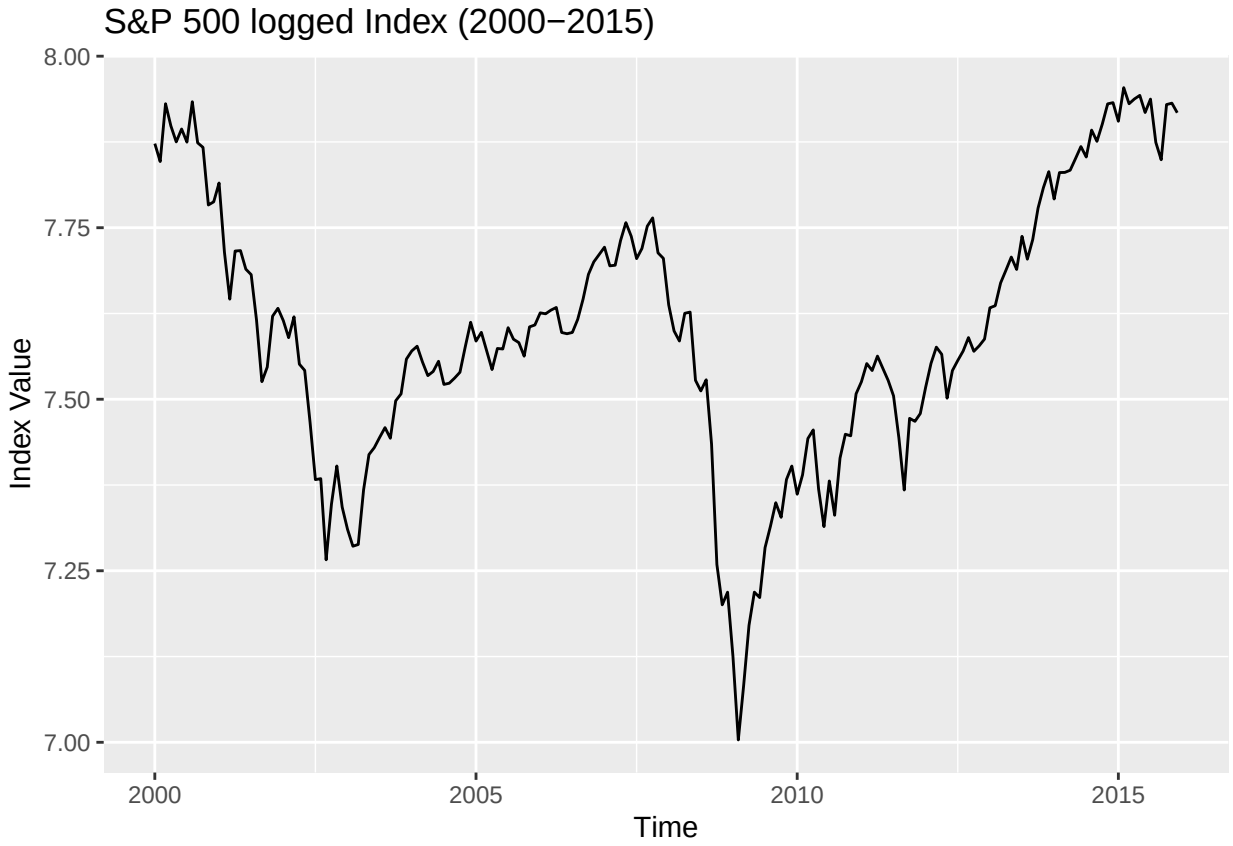



```
adf_test_result <- adf.test(SP500_real)
print(adf_test_result)
```

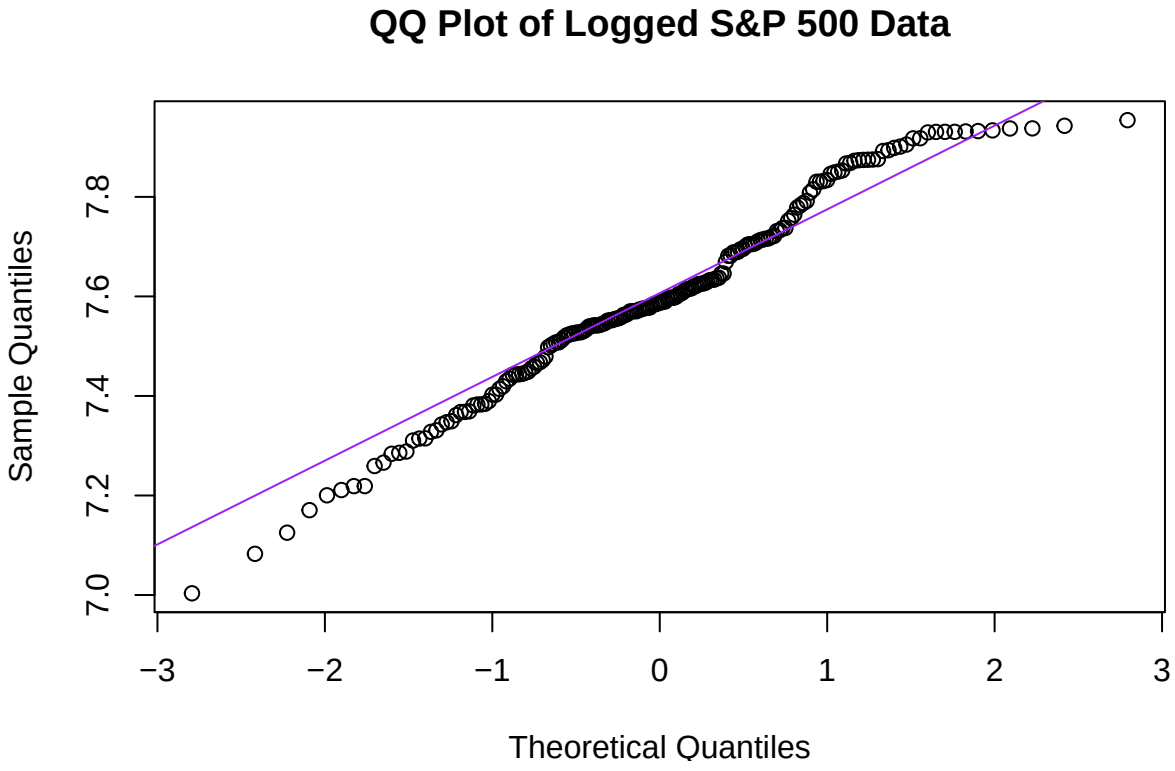
```
##
##  Augmented Dickey-Fuller Test
##
## data:  SP500_real
## Dickey-Fuller = -2.312, Lag order = 5, p-value = 0.4456
## alternative hypothesis: stationary
```

There is a heavy tail in the QQ plot, wo need to do a log or Box Cox power transformation to make data fit well.

```
# Step 2
sp500_log <- log(SP500_real)
autoplot(sp500_log) + ggtitle("S&P 500 logged Index (2000-2015)") + ylab("Index Value")
```

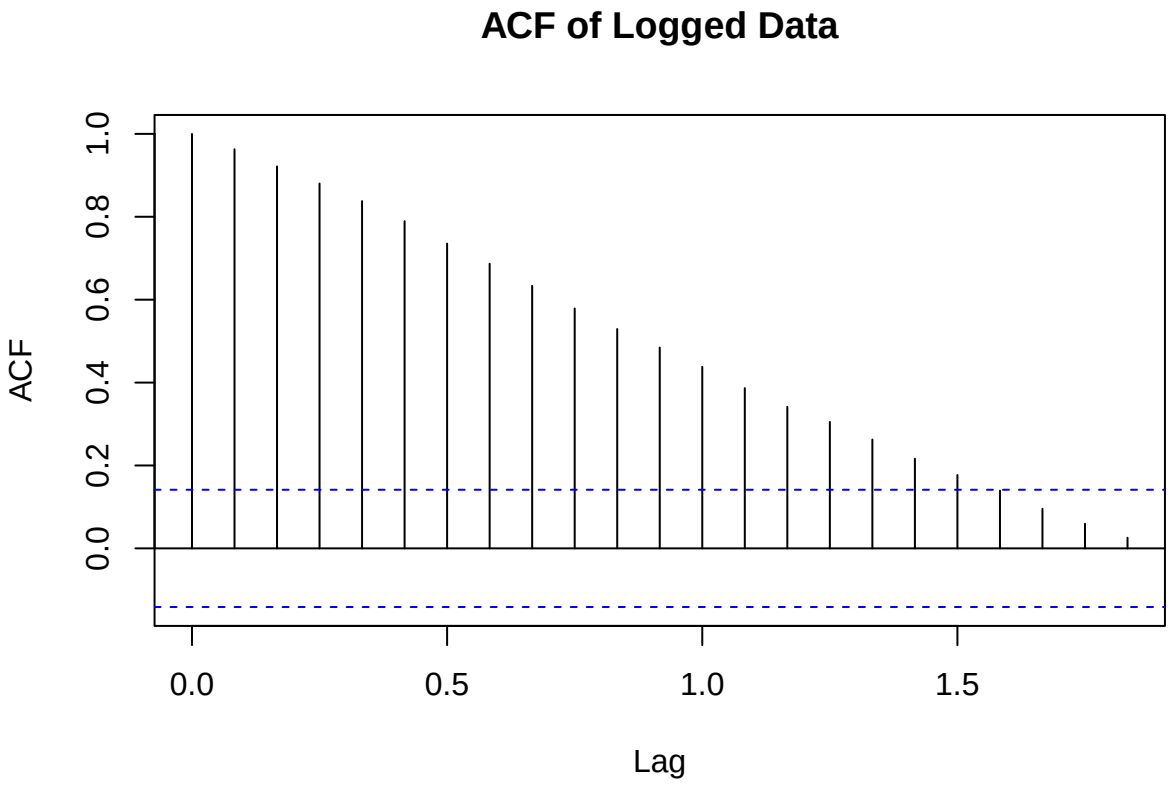



```
qqnorm(sp500_log, main="QQ Plot of Logged S&P 500 Data")
qqline(sp500_log, col="purple")
```



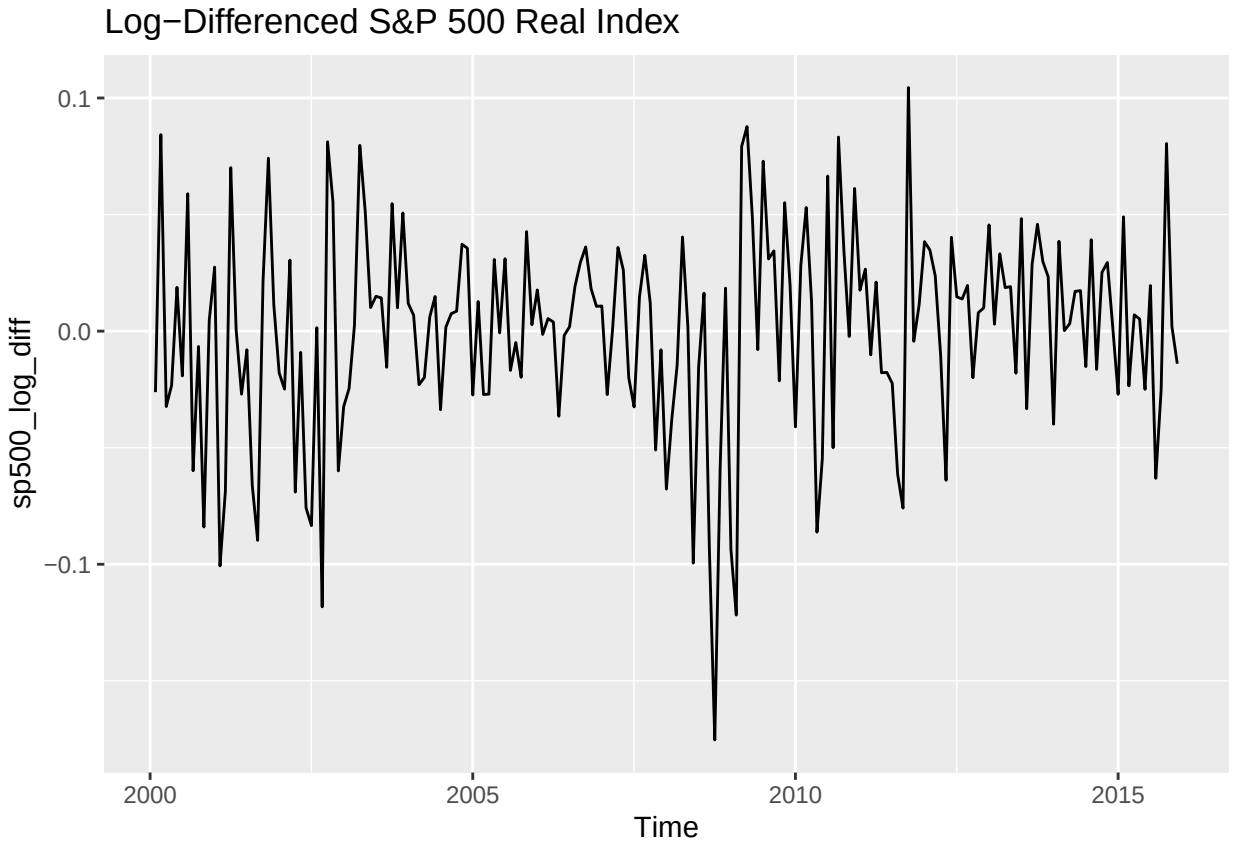
log transformed model has a better fit after that.

```r
acf(sp500_log, main="ACF of Logged Data")
```

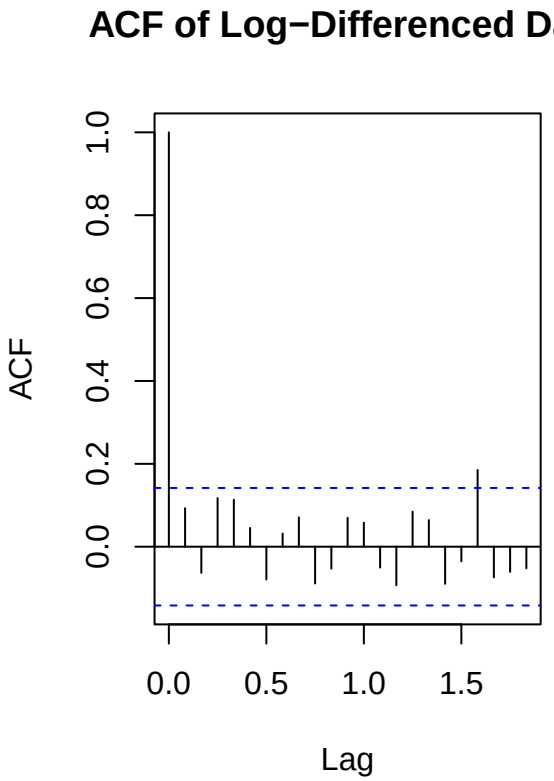


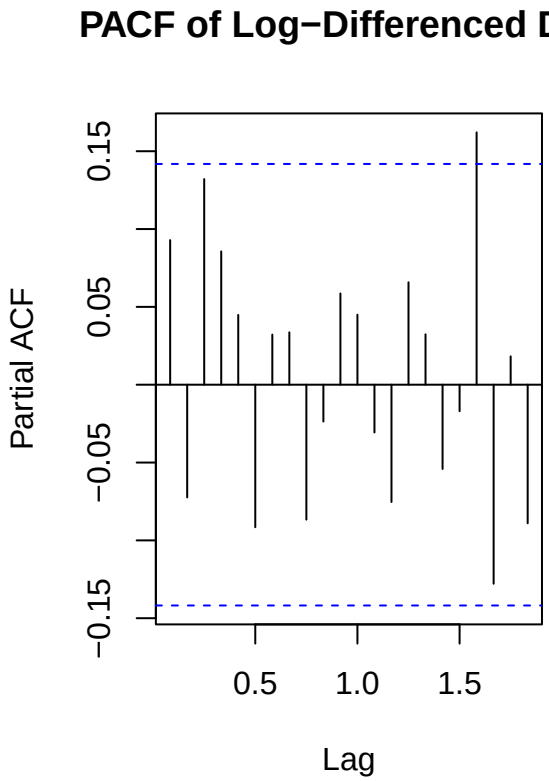
The ACF is decayed gradually, which means the data need to 1 difference level(d = 1)

```r
## Step 3
sp500_log_diff <- diff(sp500_log)
autoplot(sp500_log_diff) + ggtitle("Log-Differenced S&P 500 Real Index")
```




```
par(mfrow=c(1,2))
acf(sp500_log_diff, main="ACF of Log-Differenced Data")
pacf(sp500_log_diff, main="PACF of Log-Differenced Data")
```






```
adf_test_result <- adf.test(sp500_log_diff)
```

```
## Warning in adf.test(sp500_log_diff): p-value smaller than printed p-value
```

```
print(adf_test_result)
```

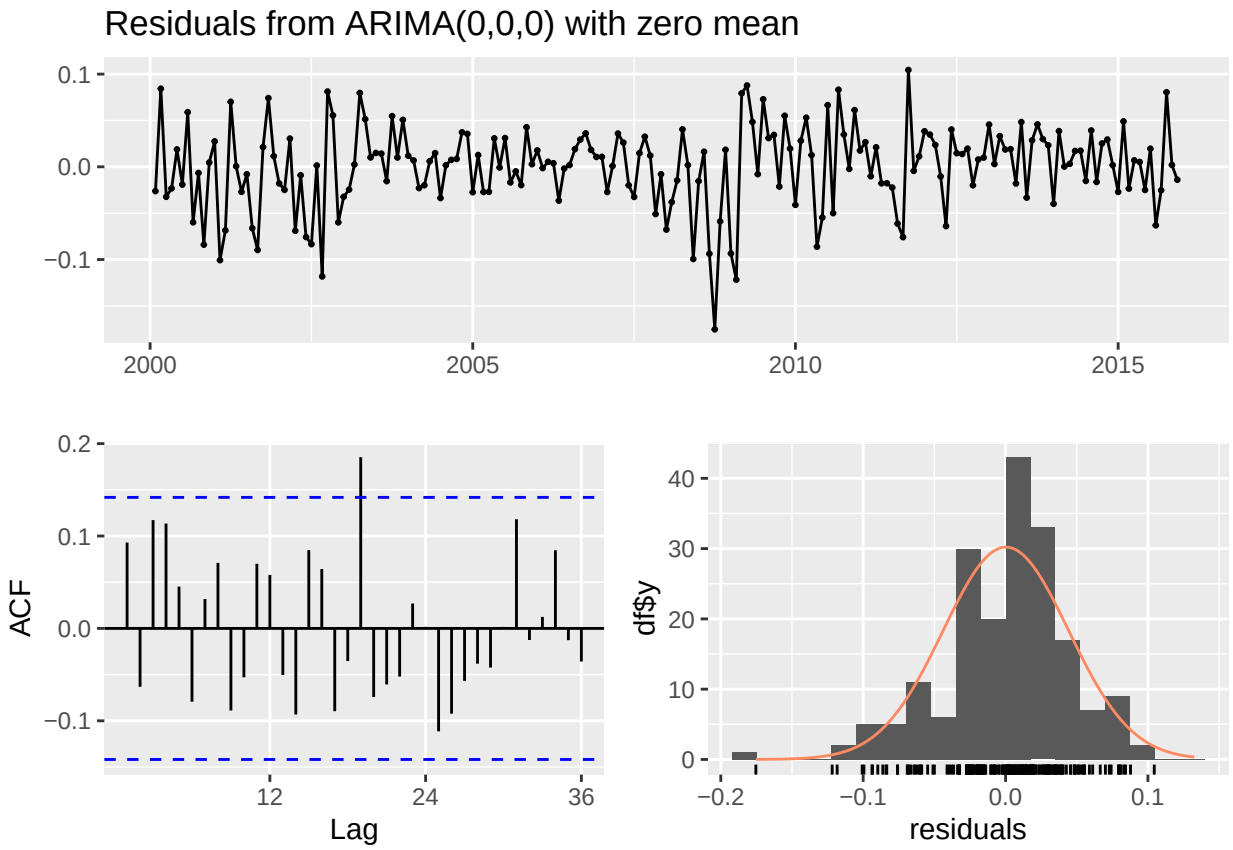
```
##
##  Augmented Dickey-Fuller Test
##
## data:  sp500_log_diff
## Dickey-Fuller = -5.306, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

The ACF and PACF are stationary now.

```
## Step 4
sarima_log_model <- auto.arima(sp500_log_diff, seasonal = FALSE)
print(summary(sarima_log_model))
```

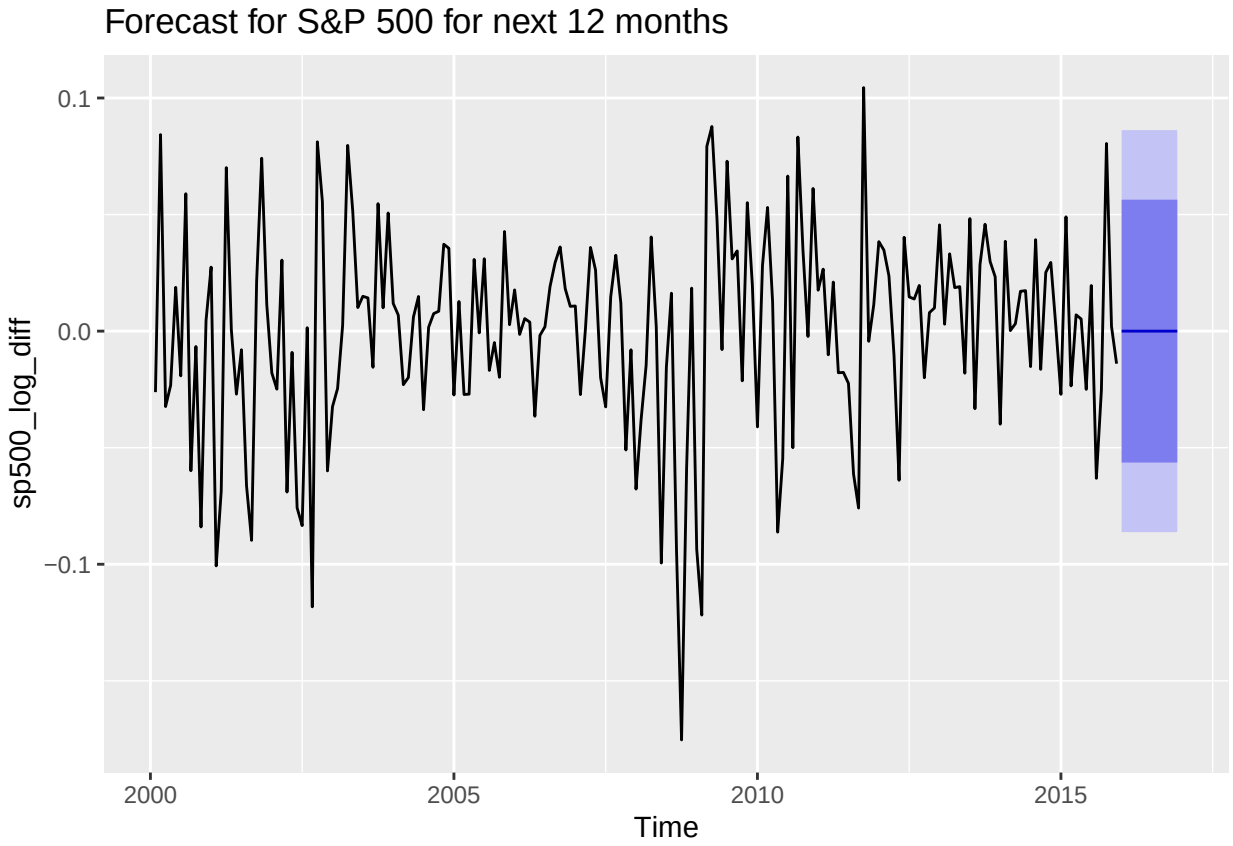
```
## Series: sp500_log_diff
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 0.001935:  log likelihood = 325.62
## AIC=-649.24   AICc=-649.22   BIC=-645.99
##
## Training set error measures:
##                        ME      RMSE        MAE MPE MAPE      MASE      ACF1
## Training set 0.0002353081 0.0439916 0.03341372 100  100 0.7298417 0.0929069
```

```
# Step 5
checkresiduals(sarima_log_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with zero mean
## Q* = 31.192, df = 24, p-value = 0.1483
##
## Model df: 0.   Total lags used: 24
```

```
forecast_values <- forecast(sarima_log_model, h=12)
autoplot(forecast_values) + ggtitle("Forecast for S&P 500 for next 12 months")
```




```
forecast_values
```

```
##          Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## Jan 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Feb 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Mar 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Apr 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## May 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Jun 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Jul 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Aug 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Sep 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Oct 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Nov 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
## Dec 2016              0 -0.05637751 0.05637751 -0.08622196 0.08622196
```

Neural network

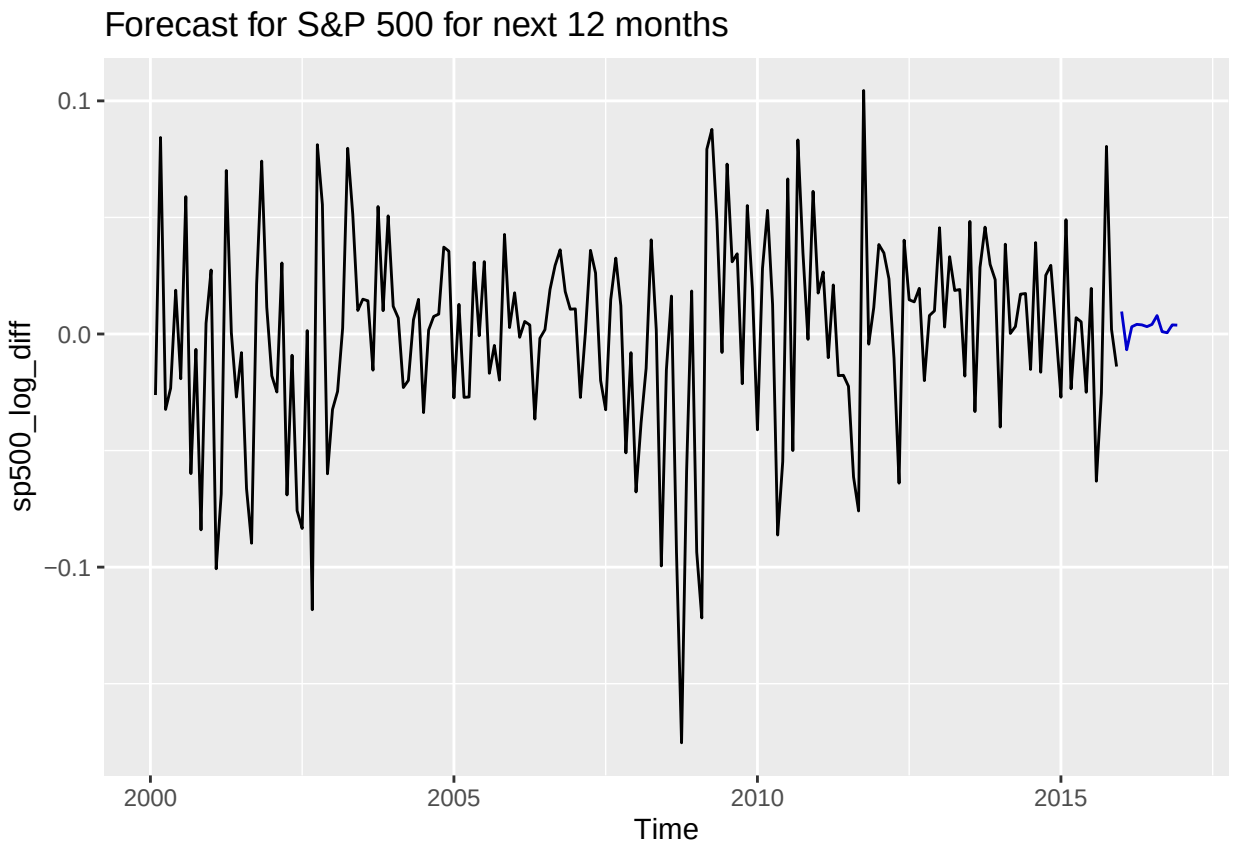
```
sp500_nn <-nnetar(sp500_log_diff)
sp500_nn
```

```
## Series: sp500_log_diff
## Model:  NNAR(3,1,2)[12]
## Call:   nnetar(y = sp500_log_diff)
##
## Average of 20 networks, each of which is
## a 4-2-1 network with 13 weights
```

```
## options were - linear output units
##
## sigma^2 estimated as 0.001223
```

```
forecasted_values_nn <- forecast(sp500_nn, h=12)

autoplot(forecasted_values_nn) + ggtitle("Forecast for S&P 500 for next 12 months")
```




```
forecasted_values_nn
```

```
##                Jan           Feb           Mar           Apr           May
## 2016  0.0096524933 -0.0067487600  0.0031012845  0.0041372841  0.0039147645
##                Jun           Jul           Aug           Sep           Oct
## 2016  0.0031653736  0.0041061120  0.0078956174  0.0010069244  0.0005492535
##                Nov           Dec
## 2016  0.0039050923  0.0037996503
```

GARCH

```
library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 4.4.3
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
```

```
##     sigma
```

```
spec <- ugarchspec(variance.model=list(garchOrder=c(1,1)),mean.model = list(armaOrder = c(0, 0)))

SP500$date <- as.Date(SP500$date)
sp500_log_diff <- xts(sp500_log_diff, order.by = SP500$date[-1])

garch_fit <- ugarchfit(spec = spec, data = sp500_log_diff[-1])
garch_fit
```

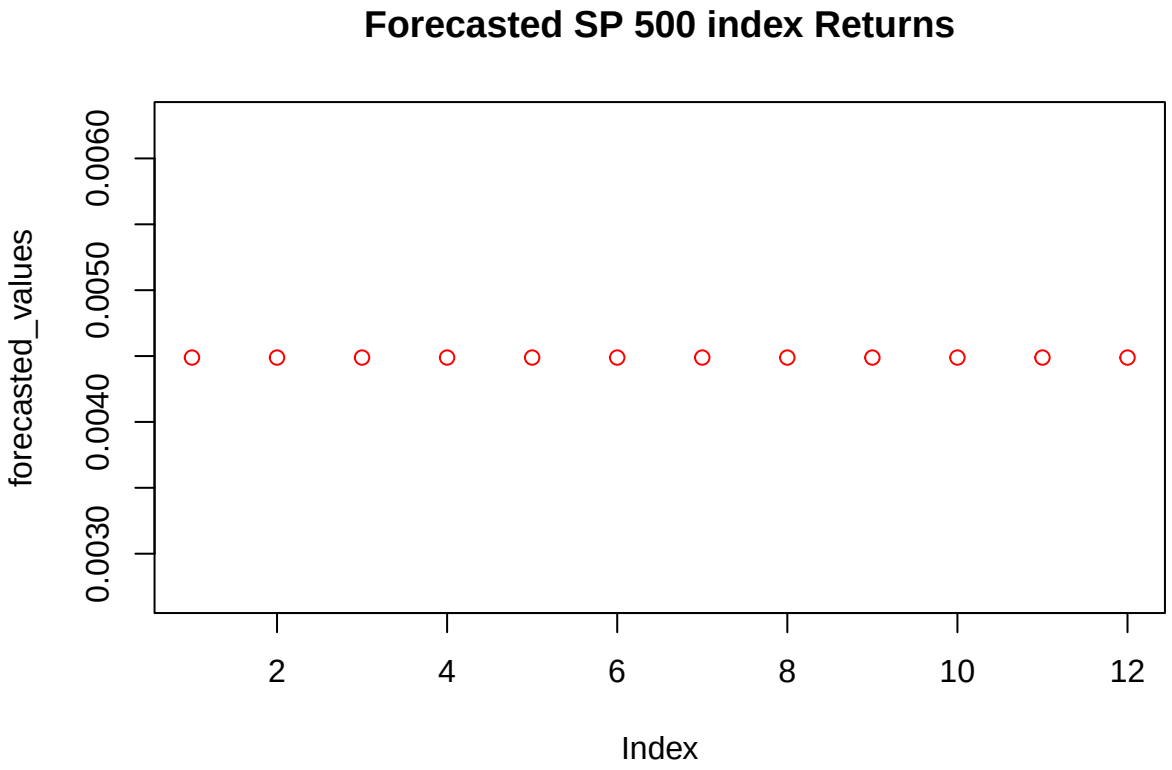
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.004489    0.002456   1.8278 0.067579
## omega   0.000083    0.000061   1.3657 0.172036
## alpha1  0.202355    0.068126   2.9703 0.002975
## beta1   0.763067    0.071119  10.7295 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.004489    0.002174   2.0646 0.038957
## omega   0.000083    0.000048   1.7129 0.086722
## alpha1  0.202355    0.057159   3.5402 0.000400
## beta1   0.763067    0.053787  14.1869 0.000000
##
## LogLikelihood : 341.2382
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.5499
## Bayes        -3.4815
## Shibata      -3.5507
## Hannan-Quinn -3.5222
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3593  0.5489
## Lag[2*(p+q)+(p+q)-1][2]    0.4569  0.7142
## Lag[4*(p+q)+(p+q)-1][5]    1.3385  0.7795
## d.o.f=0
## H0 : No serial correlation
##
```

```
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.01851  0.8918
## Lag[2*(p+q)+(p+q)-1][5]   0.66814  0.9289
## Lag[4*(p+q)+(p+q)-1][9]   1.77869  0.9297
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.0519 0.500 2.000  0.8198
## ARCH Lag[5]    1.2604 1.440 1.667  0.6575
## ARCH Lag[7]    2.0590 2.315 1.543  0.7048
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.5851
## Individual Statistics:
## mu     0.24001
## omega  0.09368
## alpha1 0.07327
## beta1  0.07969
##
## 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           0.5675 0.57107
## Negative Sign Bias  0.1812 0.85638
## Positive Sign Bias  1.6824 0.09419   *
## Joint Effect        7.6518 0.05378   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     26.63       0.1135
## 2    30     29.16       0.4569
## 3    40     44.11       0.2645
## 4    50     54.09       0.2862
##
##
## Elapsed time : 0.05454898
```

```r
forecast <- ugarchforecast(garch_fit, n.ahead = 12)

forecasted_values <- forecast@forecast$seriesFor

plot(forecasted_values, main = "Forecasted SP 500 index Returns", col = "red")
```




```
print(forecasted_values)
```

```
##       2015-12-01
## T+1  0.004488665
## T+2  0.004488665
## T+3  0.004488665
## T+4  0.004488665
## T+5  0.004488665
## T+6  0.004488665
## T+7  0.004488665
## T+8  0.004488665
## T+9  0.004488665
## T+10 0.004488665
## T+11 0.004488665
## T+12 0.004488665
```