A PROJECT REPORT

ON

Time Series Analysis on Labor Force Participation Rate and Netflix Stock Price

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Session: 2021-22

Submitted by:
Nikhil Chouhan

Under the Supervision of:
Dr. Hadi Safari Katesari

ABSTRACT

Seasonal:

Motivated by the desire to project the financial solvency of Social Security, the idea is to discuss forecasts of labor force participation rates. The purpose of this project is to give a structure to the problem of forecasting labor force participation rates, to summarize the data and to show how to implement seasonal ARIMA model. In summarizing the data, we find that differencing rates captures much of the dynamic movement of rates, as measured by short-range out-of-sample validation.

Non-Seasonal:

Stock market, a very unpredictable sector of finance, involves a large number of investors, buyers and sellers. Stock prediction has been a phenomenon since machine learning was introduced. But very few techniques became useful for forecasting the stock market as it changes with the passage of time. As time is playing a crucial rule here, Time Series (TS) analysis is used in this paper to predict short-term stock market. The first step for analyzing TS is to check whether historical stock market data is stationary using Plotting Rolling Statistics and Dickey-Fuller Test. Secondly, Trend and Seasonality is eliminated from the series to make the data a stationary series. Then, TS stochastic model known as Autoregressive Integrated Moving Average (ARIMA) is used as it has been broadly applied in financial and economic sectors for its efficiency and great potentiality for short-term stock market prediction. For comparing the performance, the three subclasses of ARIMA such as: Autoregressive (AR), Moving Average (MA), and Autoregressive Moving Average (ARMA) are also applied. Finally, the forecasted values are converted to the original scale by applying Trend and Seasonality constraints back.

INTRODUCTION

Seasonal:

The labor force participation rate represents the number of people in the labor force as a percentage of the civilian non institutional population. In other words, the participation rate is the percentage of the population that is either working or actively looking for work.
The labor force participation rate is calculated as: (Labor Force ÷ Civilian Non institutional Population) x 100. The labor force participation rate helps government agencies, financial markets, and researchers gauge the overall health of the economy.

The long-run changes in labor force participation may reflect secular economic trends that are unrelated to the overall health of the economy. For instance, demographic changes such as the aging of population can lead to a secular increase of exits from the labor force, shrinking the labor force and decreasing the labor force participation rate.

Non Seasonal:

The prediction of the stock market in time series considered one of the most challenging issues because of it volatile and noise features. The aim of this analysis is to get the accurate stocks forecasting model by comparing the results of accuracy of two customize ARIMA (p, d, q) models which will applied on Netflix stocks historical data for last five year. By applying this model to forecast for Netflix's future, especially since it showed an essential role in people's life today with what the world is facing from COVID-19. Therefore it is quite essential having a clear understanding of the present as well as forecasting the future when aiming to have a safe investment.

METHODOLOGY

Seasonal:

Predicting the future labor force participation rate using seasonal ARIMA model and customize the value of p, d, q, P, D and Q in ARIMA(p, d, q) × (P, D, Q)$_{12}$ model to get better forecasting model.

Seasonal ARIMA Model

In a seasonal ARIMA model, seasonal AR and MA terms predict x_t using data values and errors at times with lags that are multiples of *S* (the span of the seasonality).

With monthly data (and *S* = 12), a seasonal first order autoregressive model would use x_{t-12} to predict x_t. For instance, if we were selling cooling fans we might predict this August's sales using last August's sales. A seasonal second order autoregressive model would use x_{t-12} and x_{t-24} to predict x_t. Here we would predict this August's values from the past two Augusts. A seasonal first order MA(1) model (with *S* = 12) would use w_{t-12} as a predictor. A seasonal second order MA(2) model would use w_{t-12} and w_{t-24}

Non Seasonal:

Prediction the future of stocks values using ARIMA model it will be by testing the auto ARIMA values as well as build customize ARIMA (p, d, q) models to get better forecasting model. The ARIMA model applied on real Netflix stock data which is available for public on Yahoo! Finance. The dataset contains Netflix daily stock price data for five years, starting from 7 April 2015 to 7 April 2020.

ARIMA Model

ARIMA Model Auto Regressive Integrated Moving Average (ARIMA) is a model describes time series given based on observed value which can be used to forecast future values. Appling ARIMA models on any time series show patterns with no random white noise and non-seasonal. The model introduced by Box and Jenkins in 1970. To generate short-term forecasts, ARIMA models showed efficient capability outperformed complex structural models. The future value of a variable in ARIMA model is a combination of linear to the past values and errors, expressed as follows:

$Yt=\phi0+\phi1Yt-1+\phi2Yt-2+\ldots+\phi pYt-p+\varepsilon t-\theta1\varepsilon t-2-\ldots-\theta q\varepsilon t-q$. Where, Yt is the actual value and Et is the random error at t,ϕi and θj are the coefficients, p and q are integers that are often referred to as autoregressive and moving average, respectively.

DATA DESCRIPTION

Seasonal Dataset:

The Labor Force Participation Rate is collected in the CPS and published by the BLS. It is provided on a monthly basis, so this data is used in part by macroeconomists as an initial economic indicator of current labor market trends. The data set consisting “Labor Force Participation Rate” from “January 2010” to “January 2020”.

Frequency: Monthly

Link of Dataset: - https://fred.stlouisfed.org/series/LNU01300000

	DATE <chr>	LNU01300000 <dbl>
1	1992-01-01	65.7
2	1992-02-01	65.8
3	1992-03-01	66.0
4	1992-04-01	66.0
5	1992-05-01	66.4
6	1992-06-01	67.6

6 rows

Non Seasonal Dataset:

The data is taken form Yahoo Finance, it is a NFLX stock price. Basically the data consist of Open, High, Low, Close, Volume and Adj Close. But here I only using Adjusted Close to keep things simple. The data set consisting “Netflix Adjusted Close Price” from “ January 2017” to “March 2022”.

Frequency: Daily

Link of Dataset:- https://fred.stlouisfed.org/series/LNU01300000

	Date <chr>	Open <dbl>	High <dbl>	Low <dbl>	Close <dbl>	Adj.Close <dbl>	Volume <int>
1	2017-01-17	135.04	135.40	132.09	132.89	132.89	12220200
2	2017-01-18	133.21	133.65	131.06	133.26	133.26	16168600
3	2017-01-19	142.01	143.46	138.25	138.41	138.41	23203400
4	2017-01-20	139.36	140.79	137.66	138.60	138.60	9497400
5	2017-01-23	138.65	139.49	137.31	137.39	137.39	7433900
6	2017-01-24	138.11	140.93	137.03	140.11	140.11	7754700

6 rows

ANALYSIS AND RESULT

Seasonal:

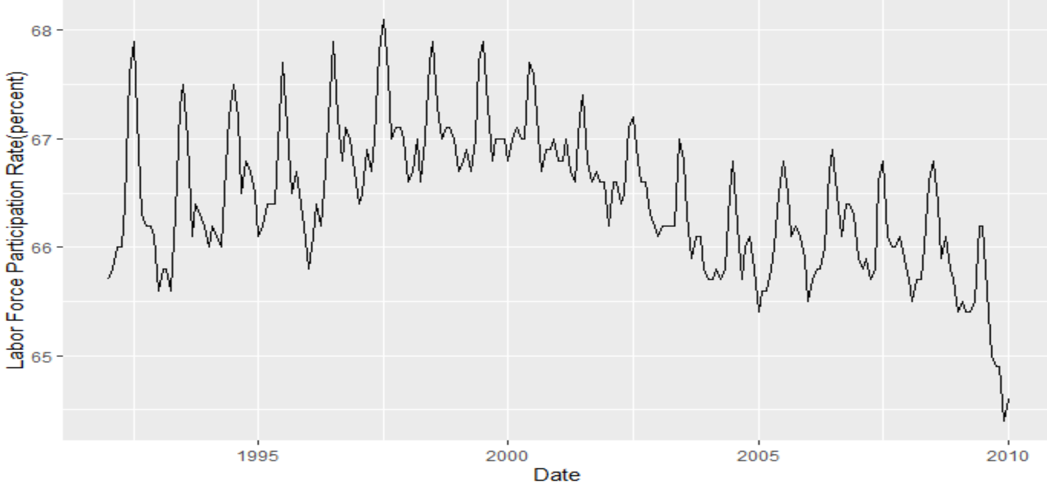
Time Series Plot of Data

This plot demonstrates the monthly Labor Force Participation rates from January 1992 through December 2009. From the below time series plot, we can see some upward trend from 1992 to 2000 and a downward trend from 2000 to 2009 but also a seasonality. We can see better and with the more details in next plot.

labor<-read.csv('C:\\Users\\HP\\Downloads\\LNU01300000 (1).csv')

labor_data=ts(data=labor$LNU01300000,frequency = 12,start = c(1992,1),end = c(2009,12))

ggplot(data=labor,aes(x=DATE,y=LNU01300000))+geom_line()+xlab('Date')+ylab('Labor Force Participation Rate(percent)')

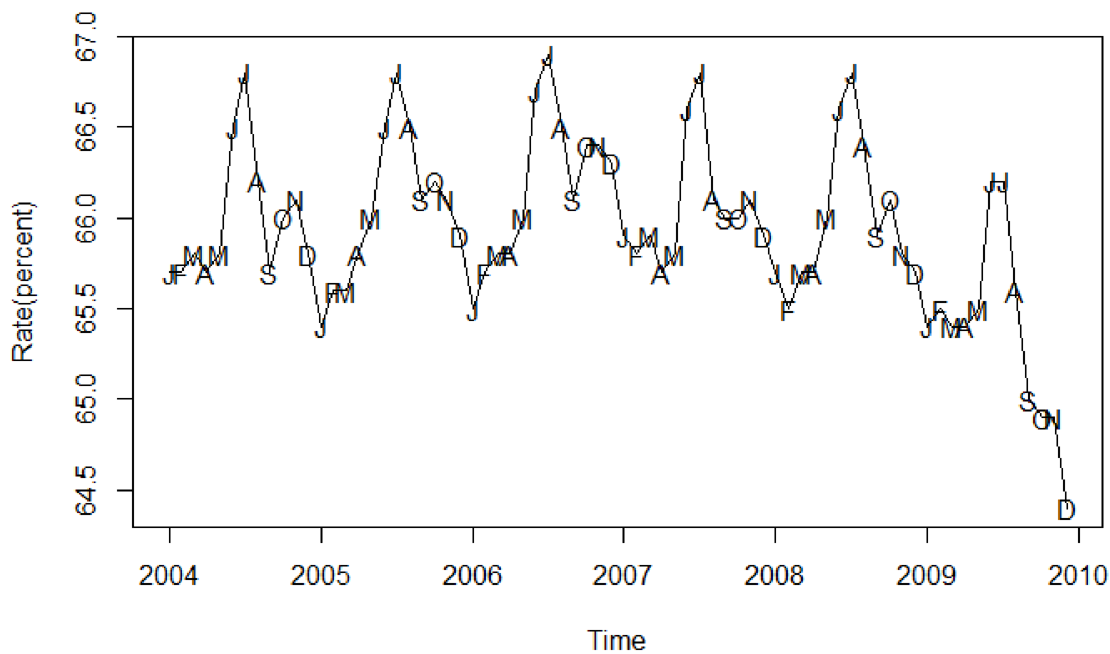


From the plot, the labor force participation rates are lower in the starting months (especially between January to May) and much higher in the months of June and July. We can consider deterministic seasonal models such as linear model but I found out that such models do not explain the behavior of this time series.

```
plot(window(labor_data,start=c(2004,1)),ylab='Rate(percent)')
```

```
Month=c('J','F','M','A','M','J','J','A','S','O','N','D')
```

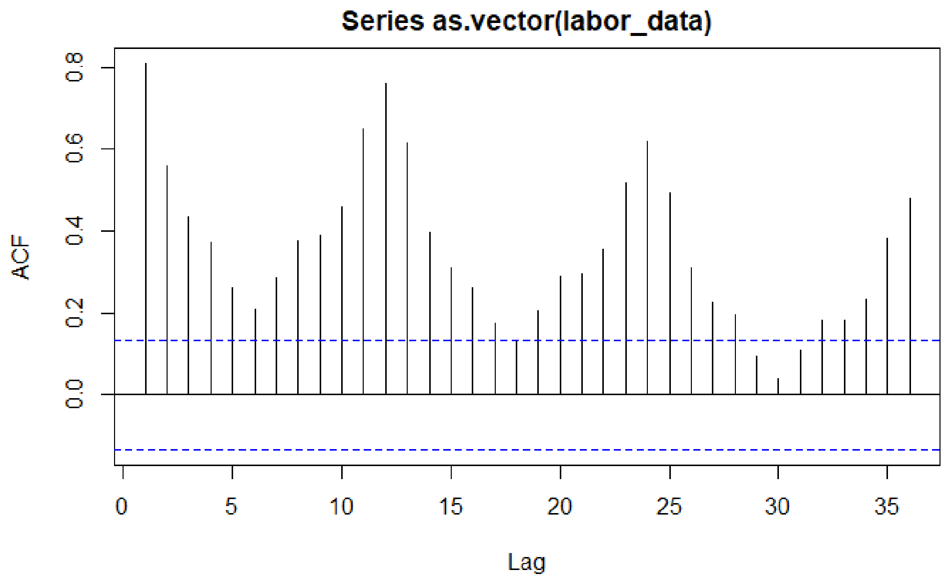
```
points(window(labor_data,start=c(2004,1)),pch=Month)
```



Model Specification

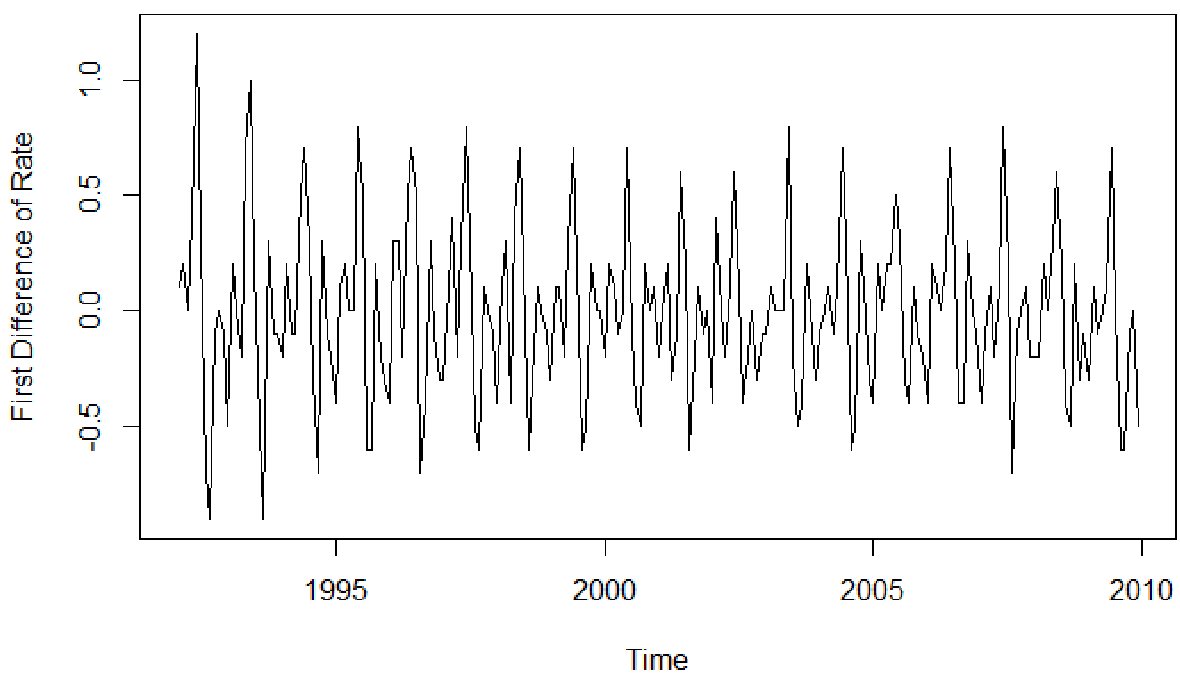
The plot demonstrates the autocorrelation for our series. In this plot, we can see the seasonal autocorrelation relationships quite prominently. The strong correlation at lags 12, 24, 36, and etc. Also, there are other correlation that needs to be modeled. To make our model stationary we need to do differencing.

```
acf(as.vector(labor_data),lag.max =36)
```



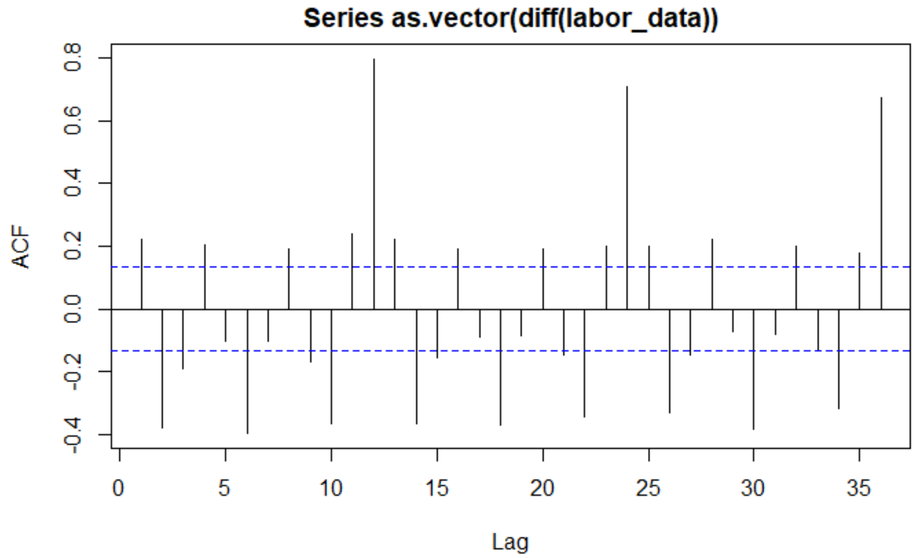
The below plot depicts the labor force participation rate after taking the first difference. The general upward trend has now disappeared.

plot(diff(labor_data),ylab='First Difference of Rate',xlab='Time')



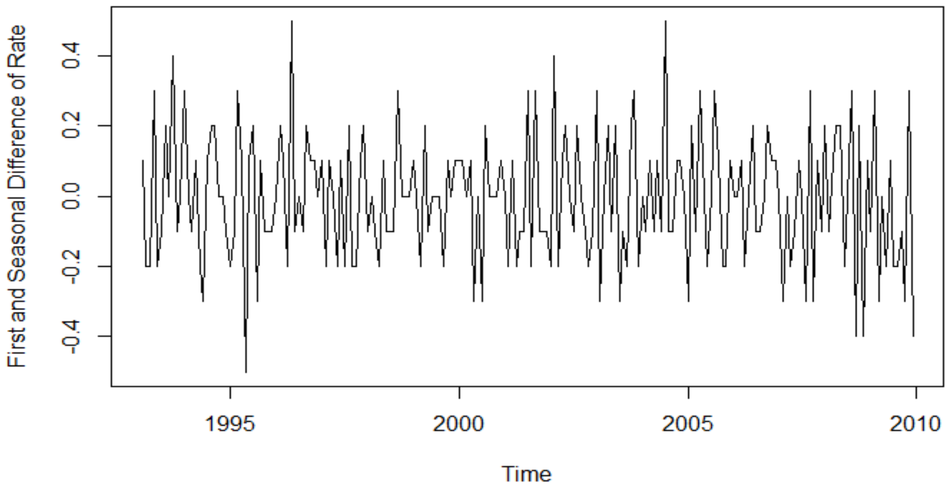
The acf plot of the first difference series show that strong seasonality still present. Perhaps seasonal differencing will bring us to a series that may be modeled parsimoniously.

acf(as.vector(diff(labor_data)),lag.max=36)



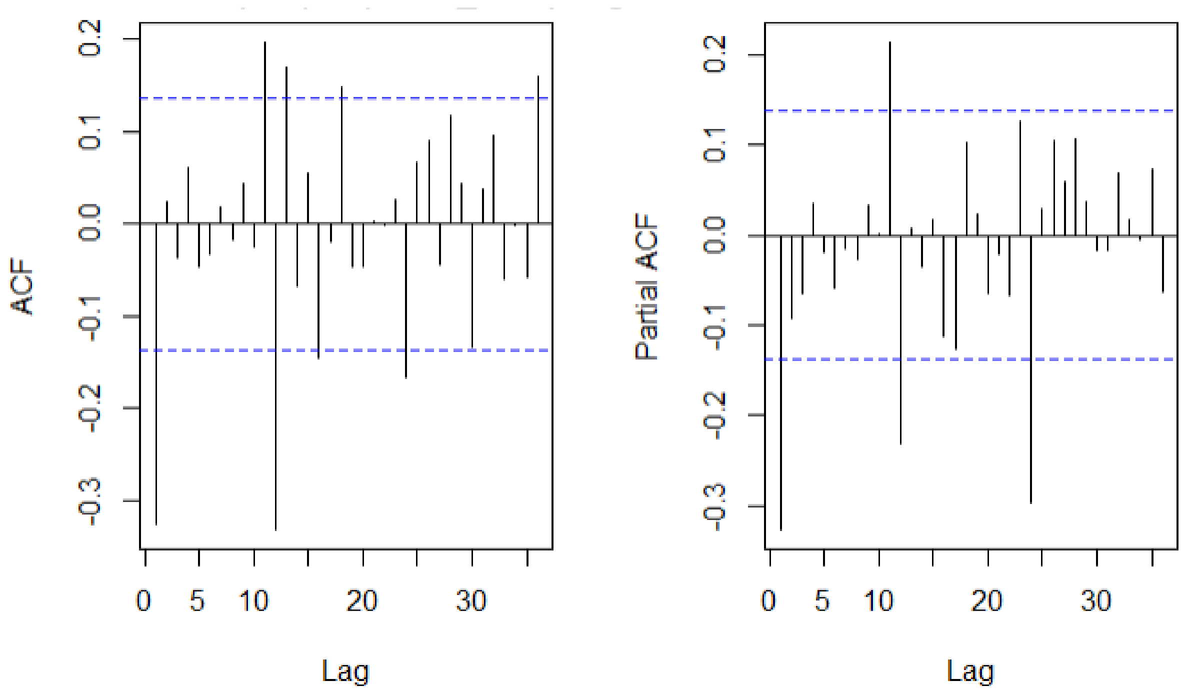
The plot demonstrates the time series plot of the Labor Force Participation Rate after taking both a first difference and a seasonal difference. It seems that almost seasonality and upward and downward trend is disappeared and we have a better series now in compare with the previous plot.

plot(diff(diff(labor_data),lag = 12),xlab='Time',ylab='First and Seasonal Difference of Rate')



The ACF plot shows that very little autocorrelation remains in the series after two differences have been taken. This plot also suggests a strong correlation at lags 1, 11, 12 and 24. Here the lag at 1 indicate non-seasonal MA(1) terms also ACF (at lag 12 and 24) indicate seasonal MA(2) terms. Similarly the PACF plot of after two differencing shows strong correlation at lag 1, 11, 12 and 24. Here the pacf at lag 1 indicate non- seasonal AR(1) terms and ACF (at lag 12 and 24) indicate seasonal AR(2) term.

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

acf(as.vector(diff(diff(labor_data),lag=12)),lag.max = 36)

pacf(as.vector(diff(diff(labor_data),lag=12)),lag.max = 36)
```



Now, we will consider specifying the multiplicative, seasonal ARIMA (1, 1, 1) × (2, 1, 2)$_{12}$ model. Before fitting our suggested models, we need to perform on our series after seasonal and non- seasonal difference by “ Dicky Fuller Unit-Root Test”.

```
adf.test(diff(diff(labor_data),lag = 12))
```

```
p-value smaller than printed p-value
        Augmented Dickey-Fuller Test

data:  diff(diff(labor_data), lag = 12)
Dickey-Fuller = -6.723, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

Since, ADF test show p value less than 0.01 suggest that our model ARIMA (1, 1, 1) × $(2, 1, 2)_{12}$ is stationary.

Parameter Estimation

With our suggested model ARIMA (1, 1, 1) × $(2, 1, 2)_{12}$, we need to perform parameter estimation of our model.

arima.labor=arima(labor_data,order=c(1,1,1),seasonal=list(order=c(2,1,2),period=12))

arima.labor

```
Call:
arima(x = labor_data, order = c(1, 1, 1), seasonal = list(order = c(2, 1, 2),
    period = 12))

Coefficients:
          ar1      ma1     sar1     sar2     sma1     sma2
      -0.1753  -0.0408  -0.4656  -0.0978  -0.0307  -0.3619
s.e.   0.4037   0.4155   0.2332   0.1608   0.2298   0.2011

sigma^2 estimated as 0.02322:  log likelihood = 90.76,  aic = -169.51
```

The coefficient estimates are all highly significant, and we proceed to check further on this model.

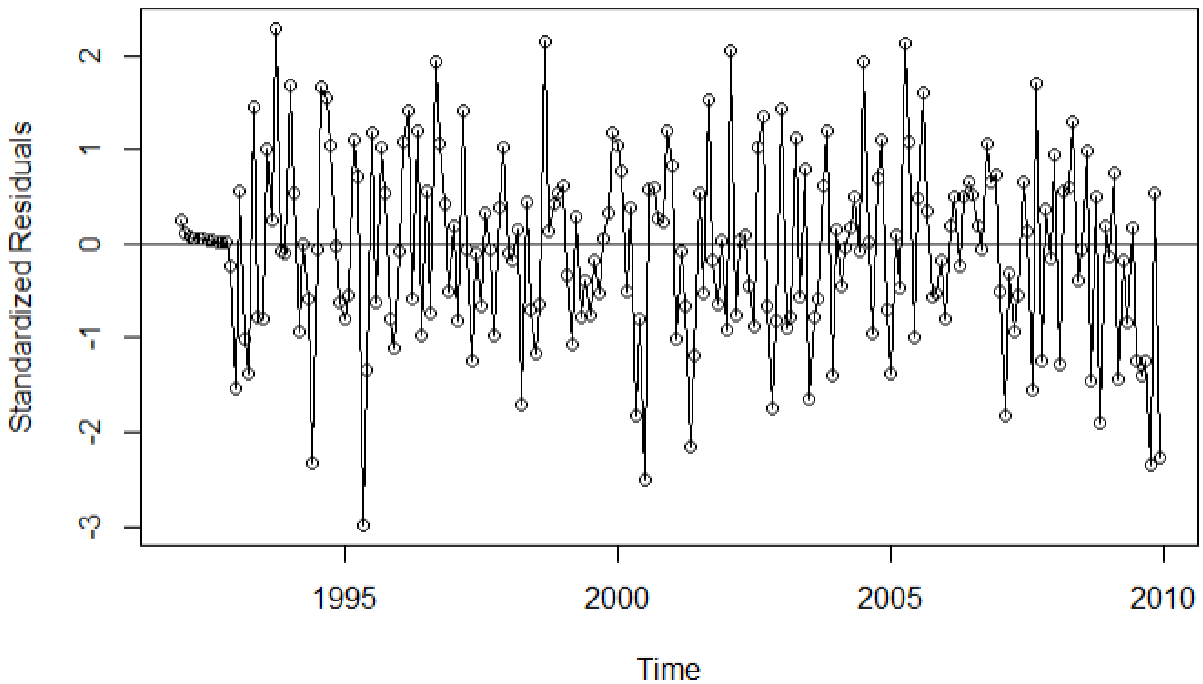
Model Diagnostics

To check the estimated the ARIMA(1,1,1)×$(2,1,2)_{12}$ model, we first look at the time series plot of the residuals.

The standardized residuals plot does not suggest any major irregularities with the model, although we need to investigate the model further for outliers.

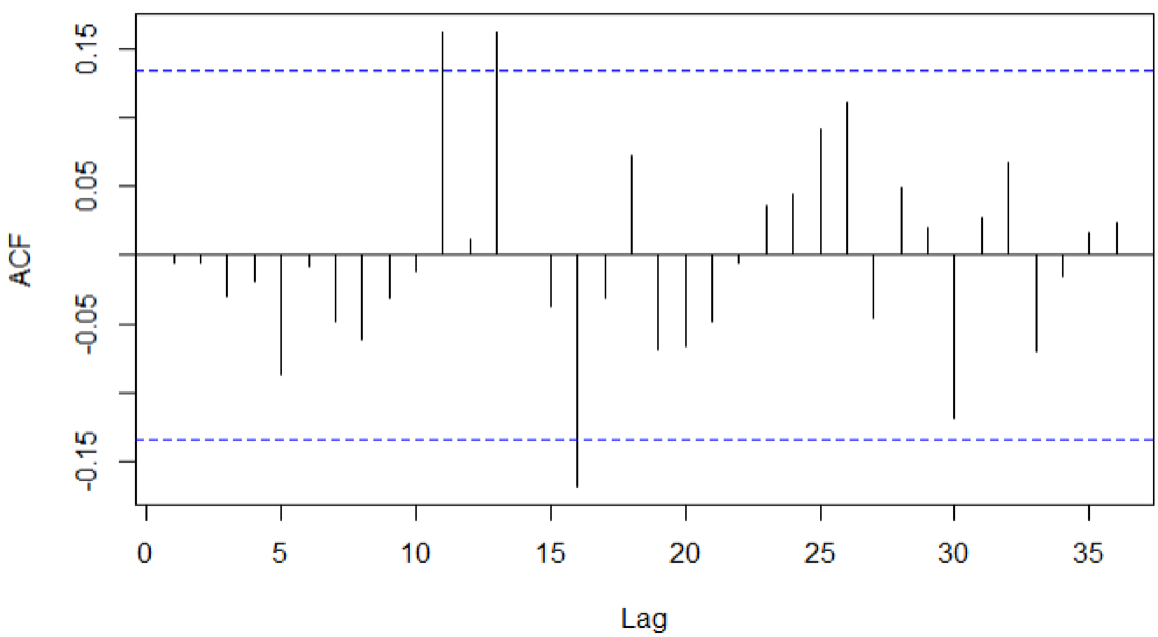
plot(window(rstandard(arima.labor),start=c(1992,1)),ylab='Standardized Residuals',type='o')

abline(h=0)



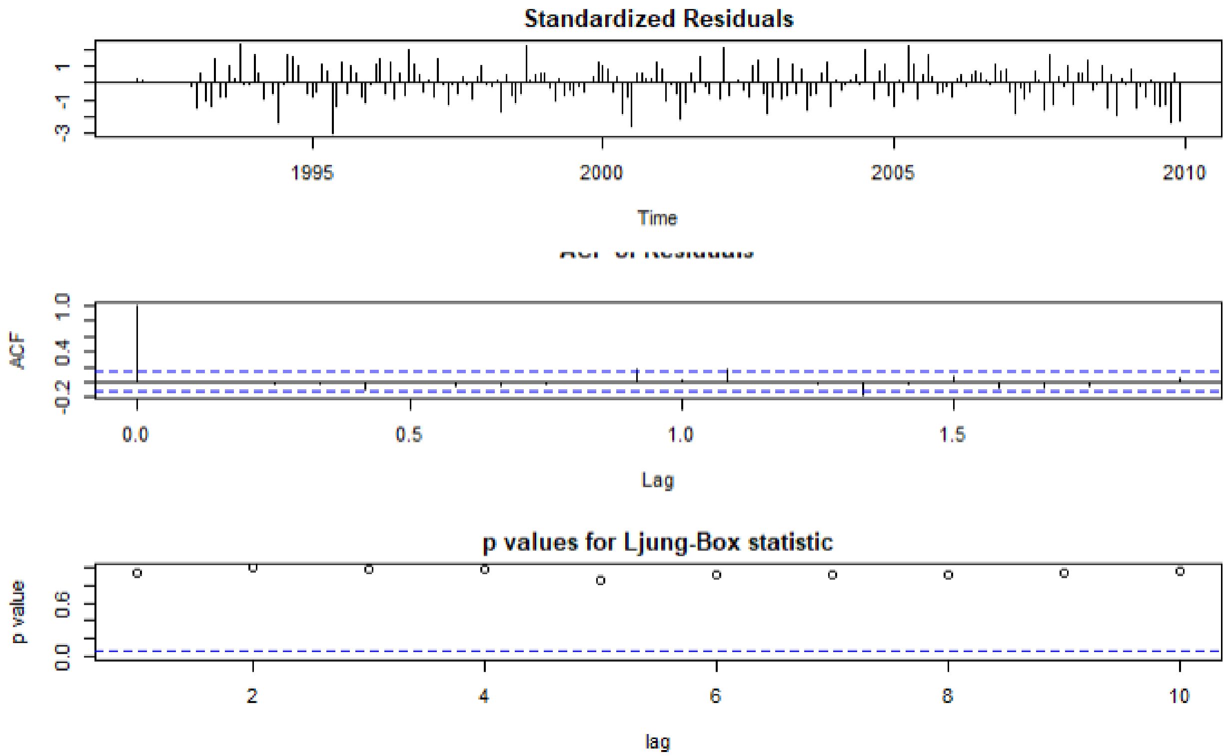
The ACF plot of the residuals shows only “statistically significant” correlation is at lag 11, 13 and 16, this correlation has a value of only 0.15, a very small correlation. Furthermore, we can think of no reasonable interpretation for dependence at lag 11, 13 or 16. Except for marginal significance at lag 22, the model seems to have captured the essence of the dependence in the series.

```
acf(as.vector(window(rstandard(arima.labor),start=c(1992,1))),lag.max=36)
```



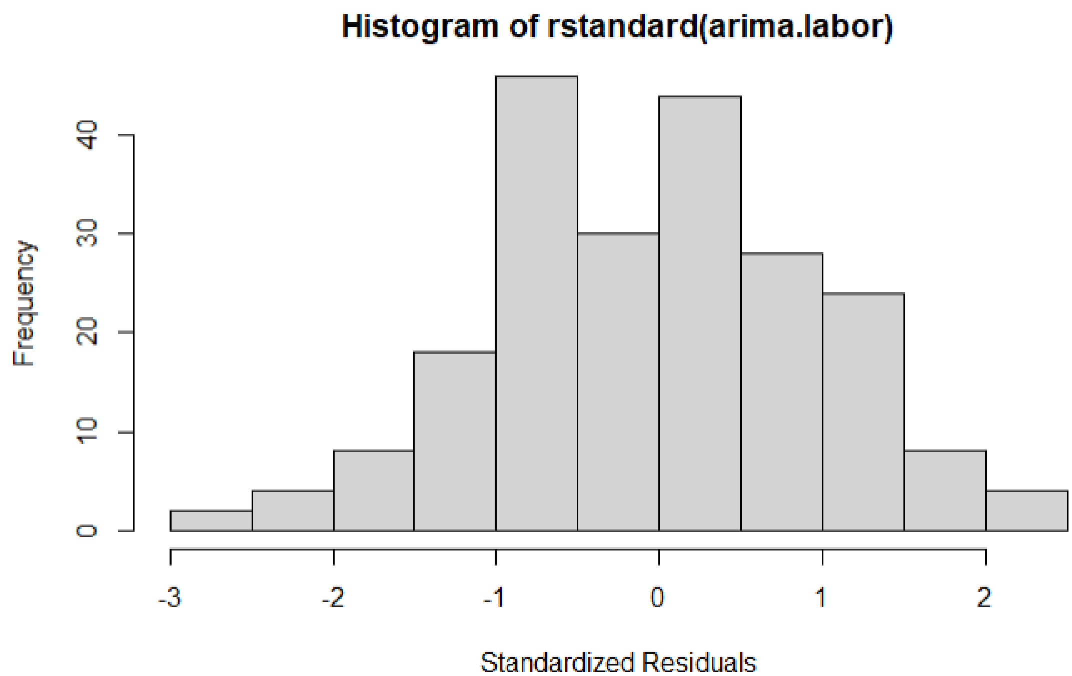
According to the Ljung-Box test for this model easily we can see a further indication that the model ARIMA (1, 1, 1) × $(2, 1, 2)_{12}$ has grabbed the dependence in the time series.

tsdiag(arima.labor)



Also, we can consider normality of the error terms by the residuals. The plot display the histogram plot of the residuals. The shape is somewhat “bell-shaped” but certainly not ideal. Perhaps we can get more information for residuals from quantile-quantile plot.

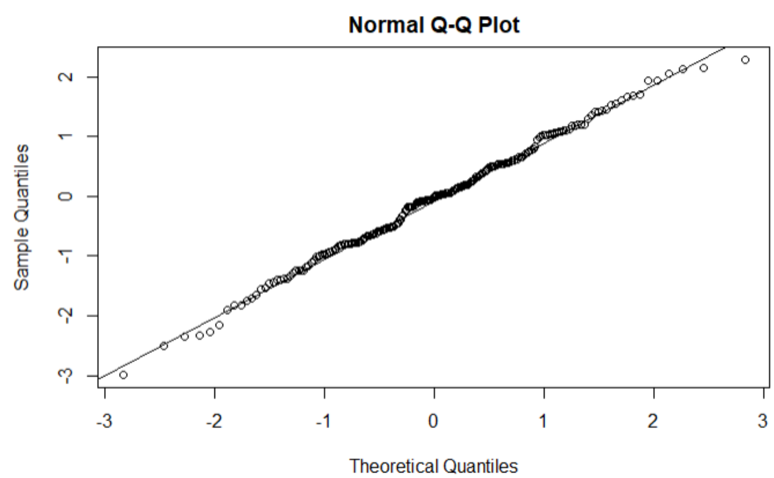
hist(rstandard(arima.labor),xlab='Standardized Residuals')



The Q-Q plot of the residuals shows two outliers in the upper tail. As a statistical test, we can use the Shapiro-Wilk test of normality.

qqnorm(rstandard(arima.labor))

qqline(rstandard(arima.labor))



According to the result of output we can see, it has a test statistic of W = 0.99517, leading to a p-value of 0.7263. Thus normality is not rejected at any of the usual significance levels. Also, we can use Ljung-Box test for this model.

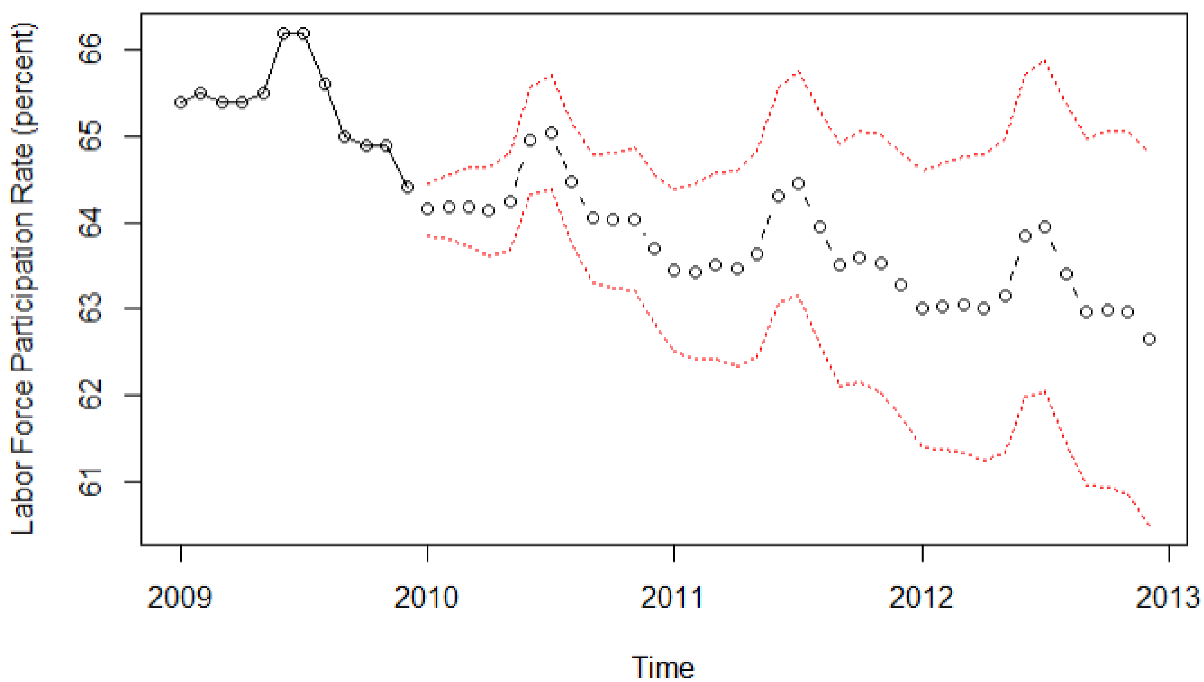
shapiro.test(residuals(arima.labor))

```
        Shapiro-Wilk normality test

data:  residuals(arima.labor)
W = 0.99517, p-value = 0.7263
```

Forecasting

The model ARIMA $(1, 1, 1) \times (2, 1, 2)_{12}$ performs well and the next step is seasonal forecasting.
The plot shows the forecasts and 95% forecast limits for a lead time of three years for the ARIMA $(1, 1, 1)\times(2, 1, 2)_{12}$ model that we fit.

plot(arima.labor,n1=c(2009,1),n.ahead=36,col = 'red',ylab='Labor Force Participation Rate (percent)')



Conclusion

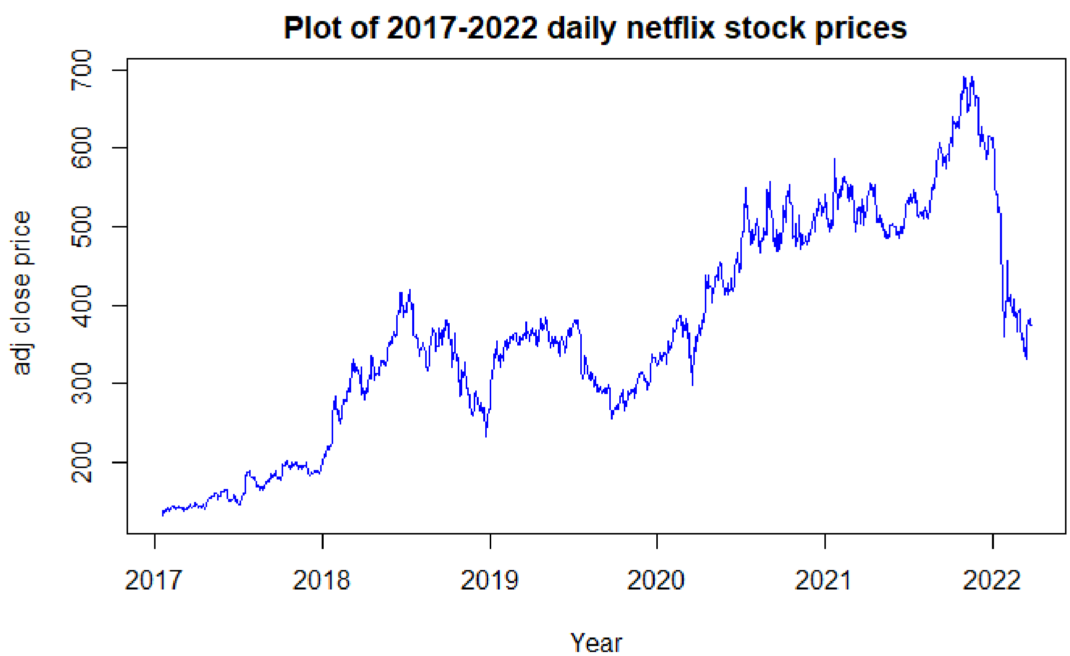
Overall, the forecast yield decent accuracy. The forecasts follow the stochastic periodicity in the data very well, and the forecast limits give a good feeling for the accuracy of the forecasts. It can be seen that the after 2009 the labor force participation rate has decreases in last three years.

Non Seasonal:

Time Series Plot of Data

The adjusted prices for the daily prices of the Netflix stock from January 1, 2017 to March 25, 2022. It can be observed that the mean value is not zero and the variance is very high. This indicates that the time series is **non-stationary with varying mean and variance**.

```
data<-read.csv('C:\\Users\\HP\\Documents\\TS project\\NFLX.csv')

data$Date<-as.Date(data$Date,format = '%Y-%m-%d')

data_mean<-aggregate(data$Adj.Close, list(data$Date), mean)

netflix<-read.zoo(data_mean, format='%Y-%m-%d')

plot(netflix,col='blue',type='l',ylab = " adj close price", main="Plot of 2017-2022 daily netflix stock prices")
```

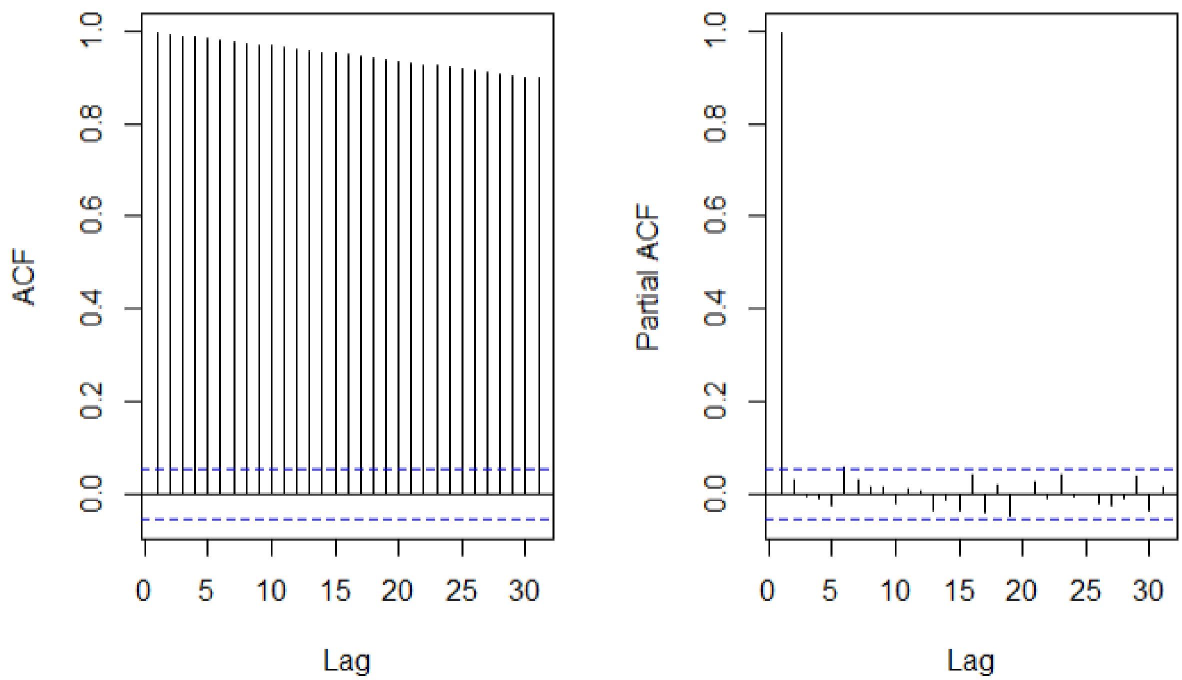


The ACF plot of non-stationary time series never dies out. As we know in AR models the ACF will dampen exponentially. The ACF is the plot used to see the correlation between the points, up to and including the lag units. We use ACF and PACF plot to identify the (q) order and the PACF will dampen exponentially. If we can note that it is a significant spike only at first lags, means that all the higher order autocorrelation is effectively explained by the first lag autocorrelation.

par(mfrow=c(1,2))

acf(coredata(netflix), main="")

pacf(coredata(netflix),main="")

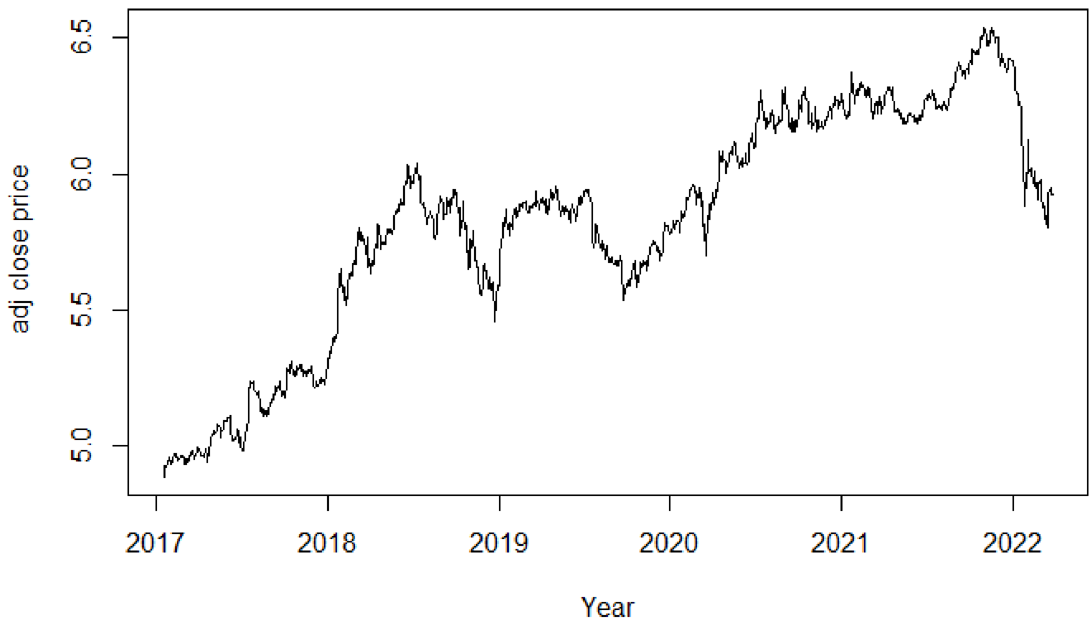


In financial time series analysis, the log transformation will scale the unit value (i.e. price in USD) of each data so it will be equally scaled (in case of multivariate time series) and the analysis will be easier. The log transformed of Netflix stock data is shown below.

netflix_log=log(netflix)

netflix_log_num=coredata(netflix_log)

plot(netflix_log,xlab = 'Year',ylab =" adj close price")



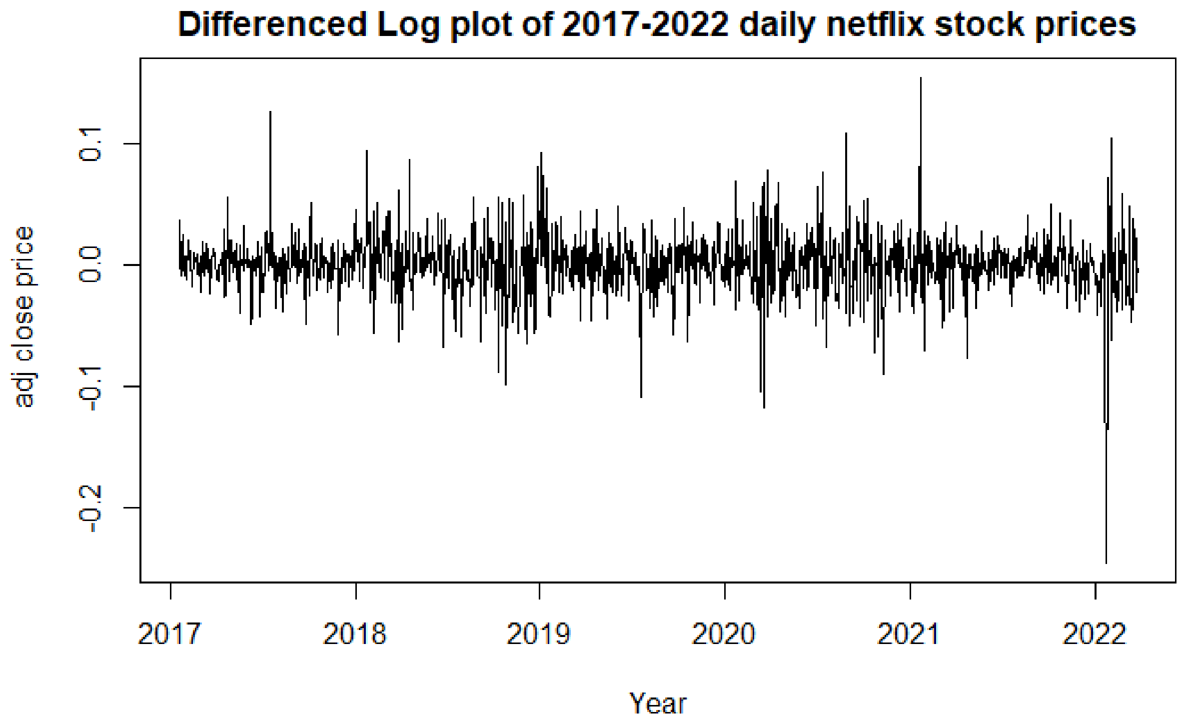
As the data has been log transformed, we can clearly see that the series shows some upward and downward trend in a given time interval. Mostly, throughout the year it was an upward trend while in mid-2018 to 2019 and mid-2021 to 2022 it has downward trend. These are the signs that the stock price movement is non-stationary.

To make data stationary, differencing method is used which helps data to transform from non-stationary to stationary time series. To stabilize the variance, the log return and one time differencing is taken. This time series plot shows that the mean is constant and nearly 0.

netflix_diff<-diff(log(netflix))

netflix_diff_num<-coredata(netflix_diff)

plot(diff(log(netflix)),xlab='Year',ylab =" adj close price", main="Log return plot of 2017-2022 daily netflix stock prices")



As the logged data has been differenced at lag 1, we can see now that the data oscillates around 0 mean, this is the main characteristic of stationary data. But, we can do another stationary testing using unit root testing. For the netflix (before differenced), it follows random walk process hence non-stationary so it must be differenced. We can test it using Augmented Dickey Fuller Test (ADF), simply explained, we test alternative hypothesis that our data is stationary against the null hypothesis that the data is non-stationary. As usual, if the resulting p-value is below 0.05, we conclude that it is significant and we can reject the null hypothesis, otherwise it has unit root and is non-stationary. From the ADF test, our logged-differenced data is stationary.

adf.test(netflix_log,k=0)

```
        Augmented Dickey-Fuller Test

data:  netflix_log
Dickey-Fuller = -1.761, Lag order = 0, p-value = 0.6795
alternative hypothesis: stationary
```

the result of ADF test on netflix_log shows that it is non-stationary as shown by the insignificant p-value of 0.6795, we can say it has a unit root. Therefore we difference it by lag 1, the ADF test for our differenced (netflix_diff) data to check if it is stationary.

adf.test(netflix_diff,k=0)

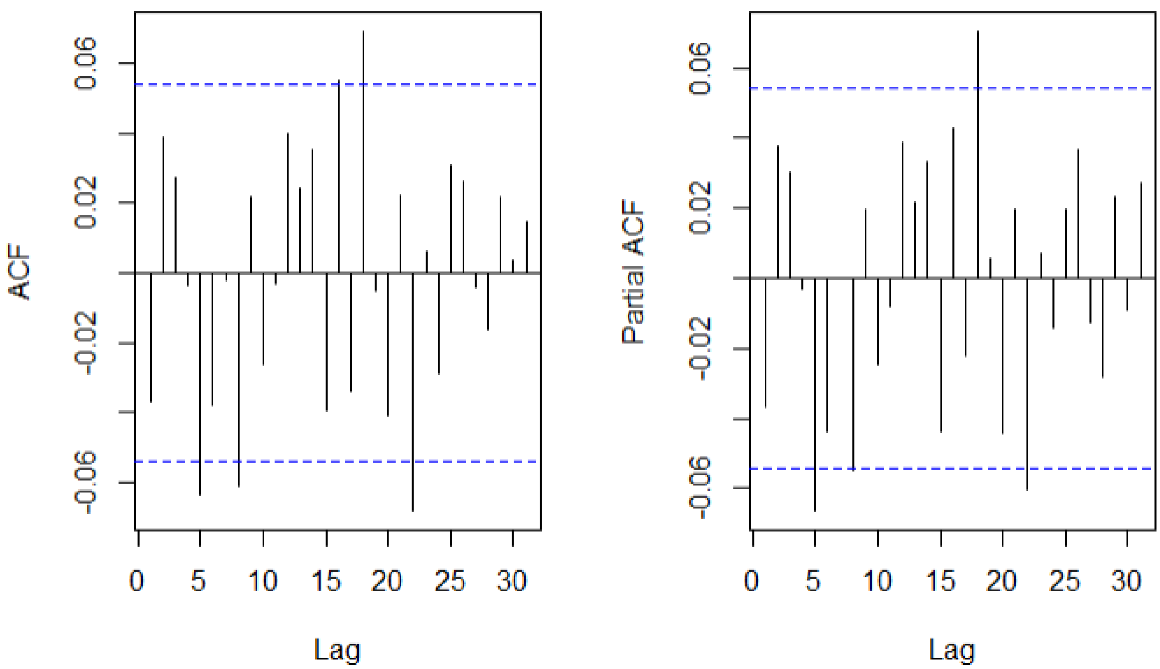
```
p-value smaller than printed p-value
        Augmented Dickey-Fuller Test

data:  netflix_diff
Dickey-Fuller = -37.53, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
```

we can see the p-value change to 0.01 (significant) and therefore we conclude that our differenced data is stationary at lag-1 (proving it is initially a random walk) and we do not have unit root in out differenced data, therefore it is appropriate for arima model.

par(mfrow=c(1,2))

acf(netflix_diff_num,main="")

pacf(netflix_diff_num, main="")



From the correlogram above, the sample ACF and PACF shows significance spike at lag 5. These suggest an MA (5) model for the difference of the log of Netflix stock price. An AR (5) model (ignoring some significant spikes at 8, 18, and 22).

Overall we could see that our data might fit ARMA(p, d, q) model given the fact it possess both ACF and PACF characteristic of AR(p) and MA(q) model given lag(d) of 1.

eacf(abs(netflix_diff_num))

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x x x x x x  x  x  x
1 x o o o o o o o o o o  o  o  o
2 x x o o o o o x o o o  o  o  o
3 x x x o o o o x o o o  o  o  o
4 x x x o o o o o o o o  o  o  o
5 x x x o x o o o o o o  o  o  o
6 x x x x x x o o o o o  o  o  o
7 o x x x x x x o o o o  o  o  o
```

Using the absolute value of the log return of daily Netflix stock price in EACF is better than square value of the log return of data due to the significant volatility of financial data. The EACF of the value of the absolute differenced log return of data suggest an ARMA (1, 1) model.

The two suggested model after analysis is ARIMA (5, 1, 5) and ARIMA (1, 1, 1). Now, we are going to perform parameter estimation to find the best suited model.

arima(netflix_log_num,order = c(5,1,5),method ='CSS')

```
Call:
arima(x = netflix_log_num, order = c(5, 1, 5), method = "CSS")

Coefficients:
         ar1     ar2     ar3      ar4      ar5      ma1      ma2      ma3     ma4      ma5
      0.2927  0.0682  0.5945  -0.5855  -0.0202  -0.3297  -0.0140  -0.5916  0.6142  -0.0962
s.e.  0.2266  0.2617  0.0825   0.1899   0.2575   0.2256   0.2654   0.0849  0.1915   0.2596

sigma^2 estimated as 0.000638:  part log likelihood = 2953.36
```

arima(netflix_diff_num,order = c(5,1,5),method='ML')

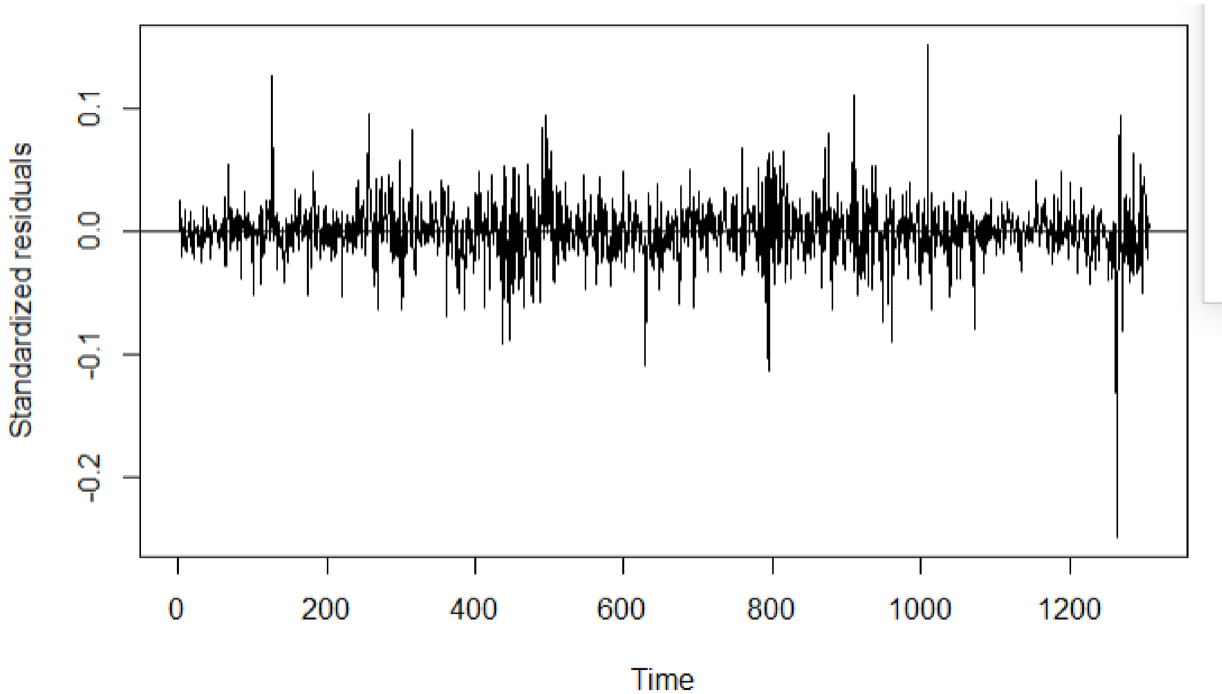
```
Call:
arima(x = netflix_diff_num, order = c(5, 1, 5), method = "ML")

Coefficients:
         ar1     ar2     ar3      ar4      ar5      ma1     ma2      ma3     ma4      ma5
      0.2657  0.1551  0.6240  -0.6049  -0.1144  -1.3017  0.2016  -0.5205  1.2505  -0.6278
s.e.  0.1869  0.0941  0.0689   0.1620   0.0288   0.1882  0.1442   0.0955  0.1760   0.1649

sigma^2 estimated as 0.0006366:  log likelihood = 2949.81,  aic = -5879.63
```

The parameter estimation for ARIMA (5, 1, 5) model using Conditional Sum of Squares and Maximum Likelihood. The coefficient estimates are highly significant, now the next step is to find the residuals of the model.

series=arima(netflix_diff_num,order = c(5,1,5))

plot(rstandard(series),ylab='Standardized residuals',type='l')

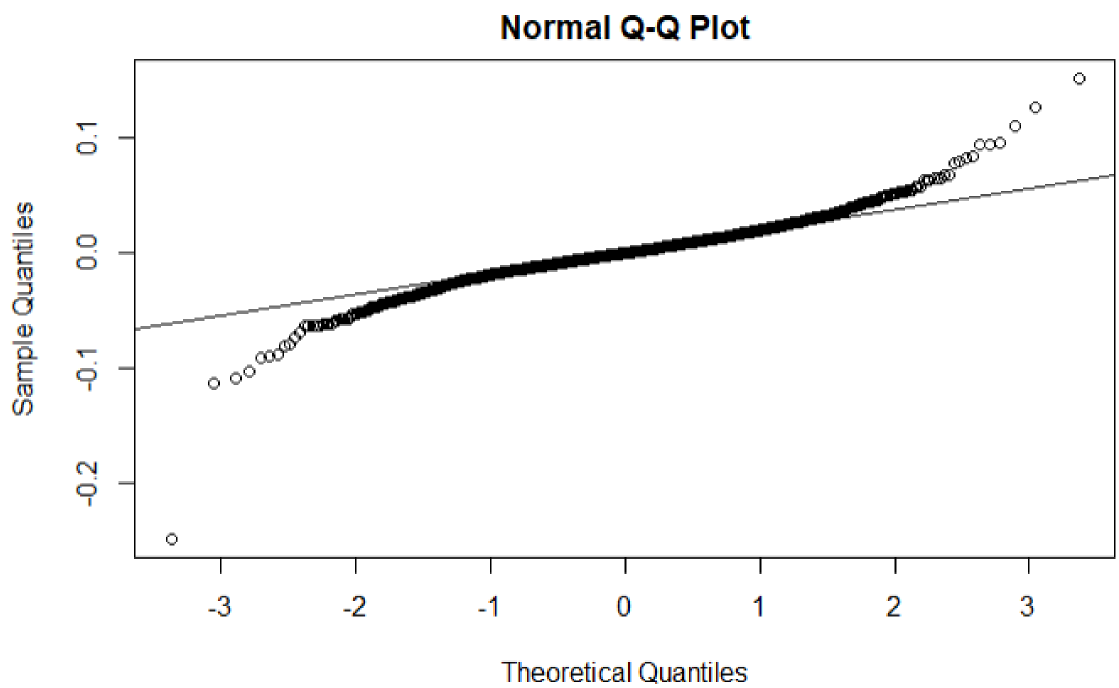
abline(h=0)



The plot for the standardization residuals form the ARIMA (5, 1, 5) model fitted to the Netflix stock price series. The parameters were estimated using maximum likelihood. Here we see possible reduced variation in the middle of the series and increased variation at the end of the series, not exactly an ideal plot of residuals.

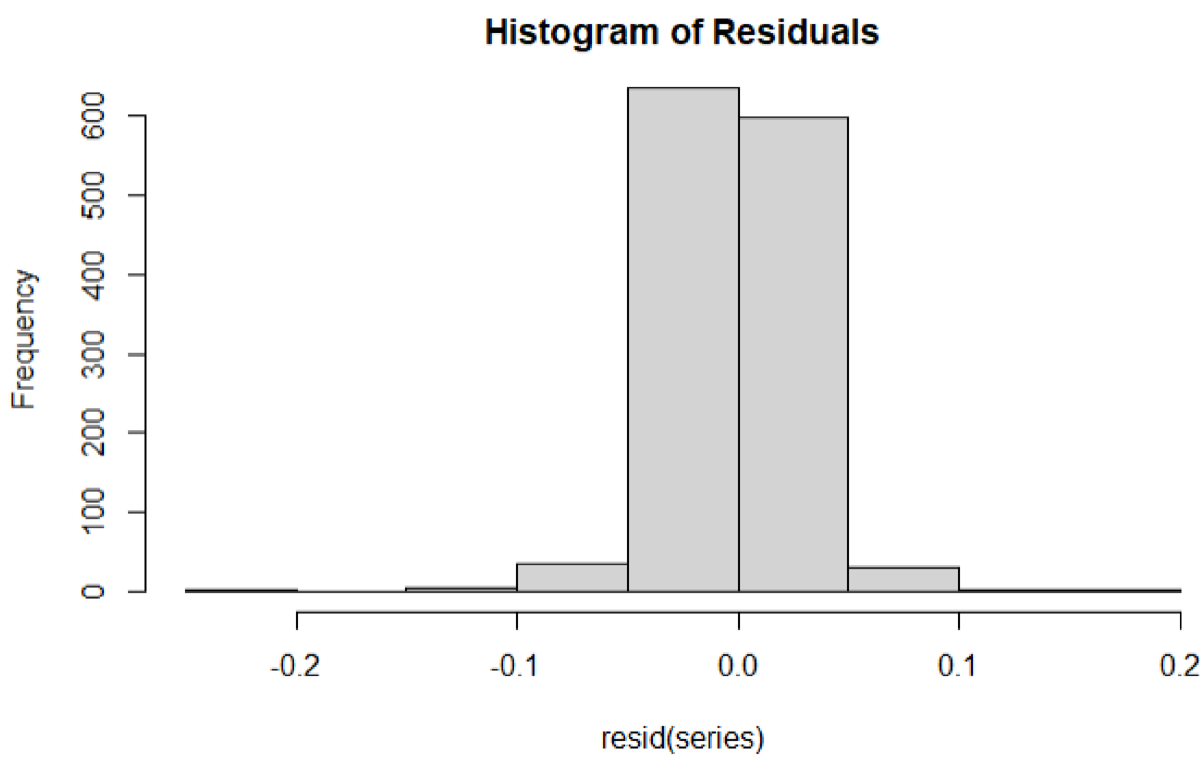
qqnorm(resid(series));

qqline(resid(series))



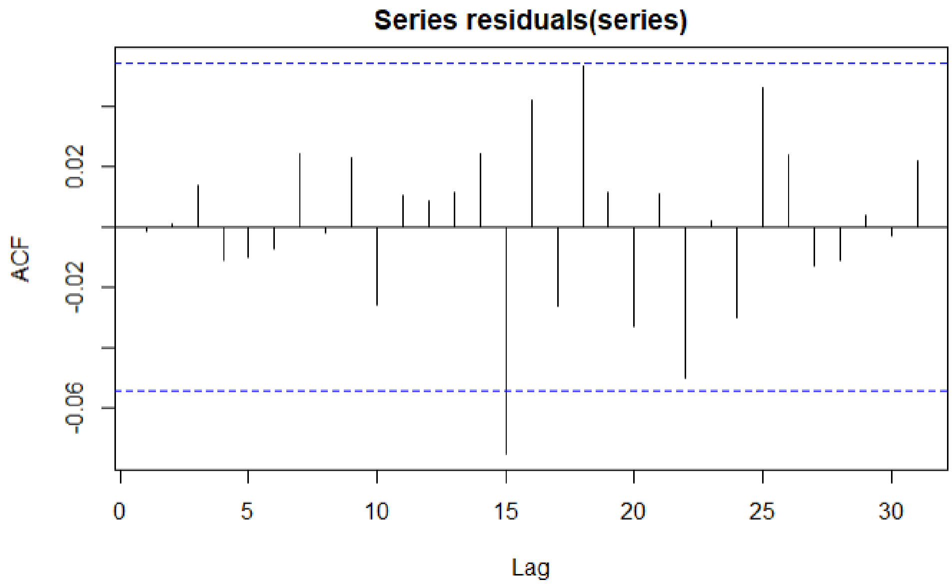
The quantile- quantile plot for the residuals from the ARIMA (5, 1, 5) model for the Netflix stock price time series. Here the extreme values look suspect.

```
hist(resid(series),main="Histogram of Residuals")
```



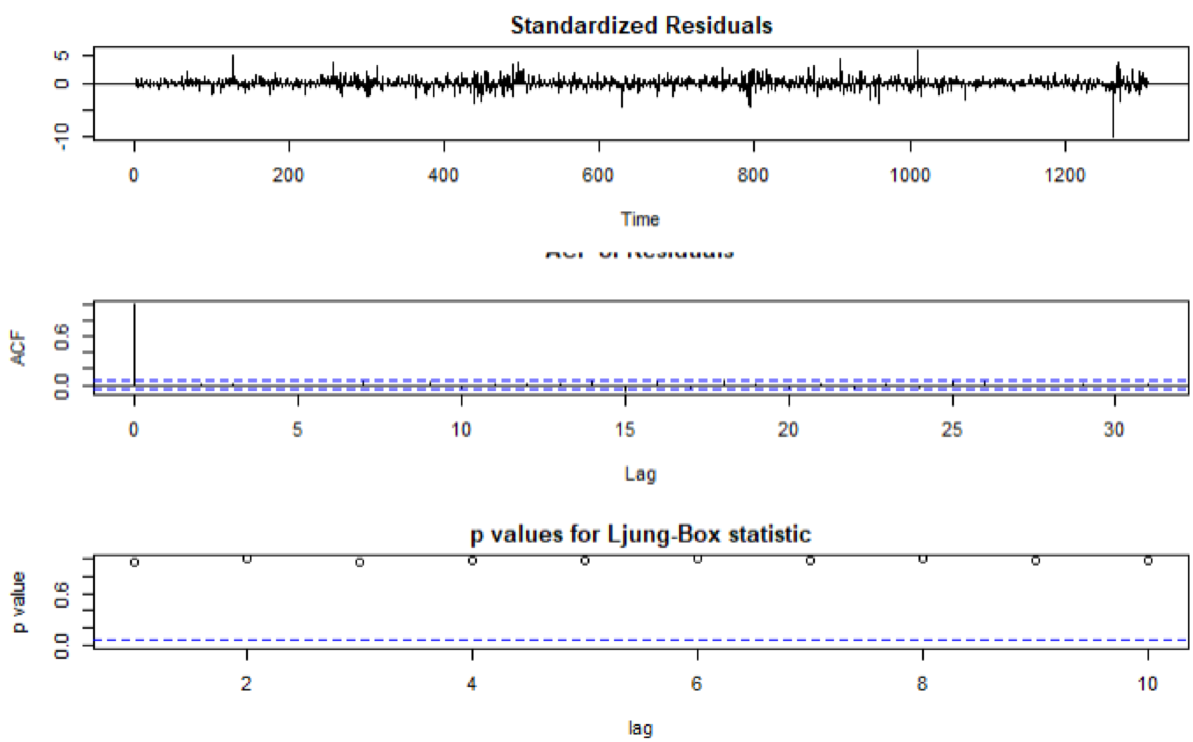
The plot demonstrates the histogram of the residuals. The shape of the plot is to some extent is bell shaped but definitely not exactly bell shaped.

acf(residuals(series),na.action = na.omit)



The sample ACF plot of the residuals does not show any statistically significant correlation here except at lag 15.

tsdiag(series)



According to the Ljung-Box test for this model easily we can see a further indication that the model ARIMA (5, 1, 5) shows p value all above 0.05. It means we cannot reject null hypothesis and our residuals are random.

Another suggested model is ARIMA (1, 1, 1) model, we need to perform all the steps and for parameter estimation of the model we have:

arima(netflix_log_num,order = c(1,1,1),method ='CSS')

```
Call:
arima(x = netflix_log_num, order = c(1, 1, 1), method = "CSS")

Coefficients:
          ar1     ma1
      -0.4598  0.4212
s.e.   0.3953  0.4044

sigma^2 estimated as 0.0006459:  part log likelihood = 2945.27
```

arima(netflix_diff_num,order = c(1,1,1),method='ML')

```
Call:
arima(x = netflix_diff_num, order = c(1, 1, 1), method = "ML")

Coefficients:
          ar1      ma1
      -0.0373  -0.9977
s.e.   0.0277   0.0035

sigma^2 estimated as 0.0006463:  log likelihood = 2939.91,  aic = -5875.81
```

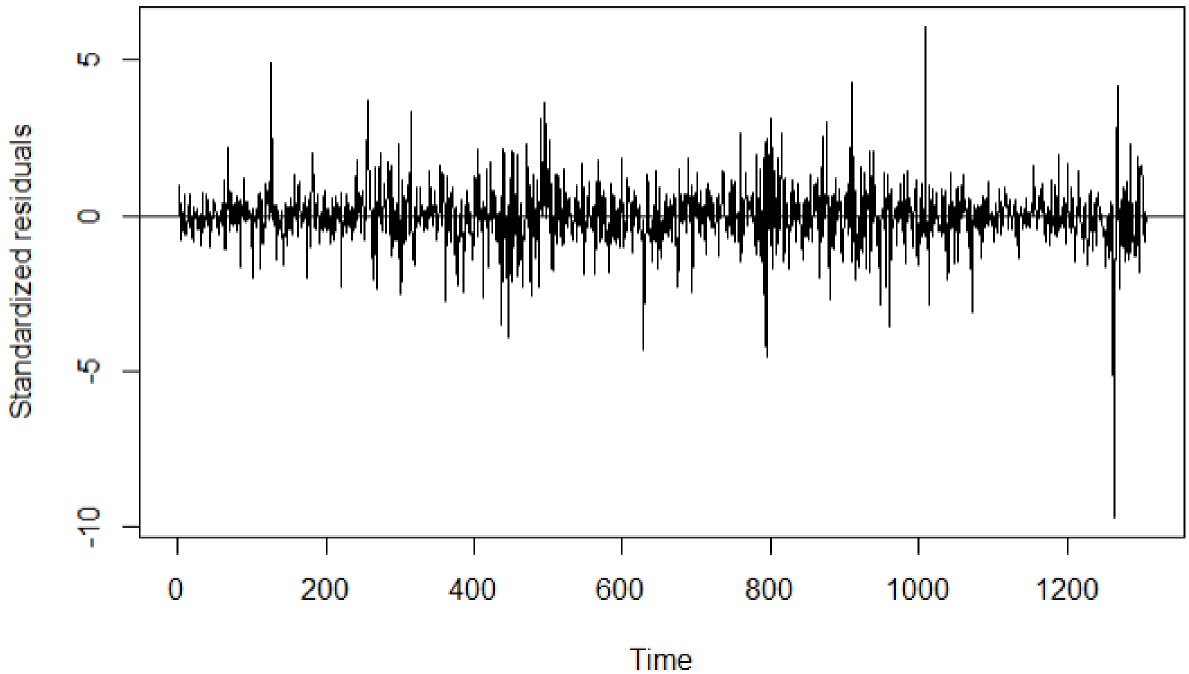
The parameter estimation for ARIMA (1, 1, 1) model using Conditional Sum of Squares and Maximum Likelihood. The coefficient estimates are highly significant for this model. As compared with first model, the log likelihood value for the first model (ARIMA (5, 1, 5)) is more than the second model (ARIMA (1, 1, 1)). Another useful criterion is AIC that the AIC of the first model is less than the second model.

The next step is to check the model standardization residuals for the ARIMA (1, 1, 1) model fitted to the Netflix stock price series.

series2=arima(netflix_diff_num,order = c(1,1,1))

plot(rstandard(series2),ylab='Standardized residuals',type='l')

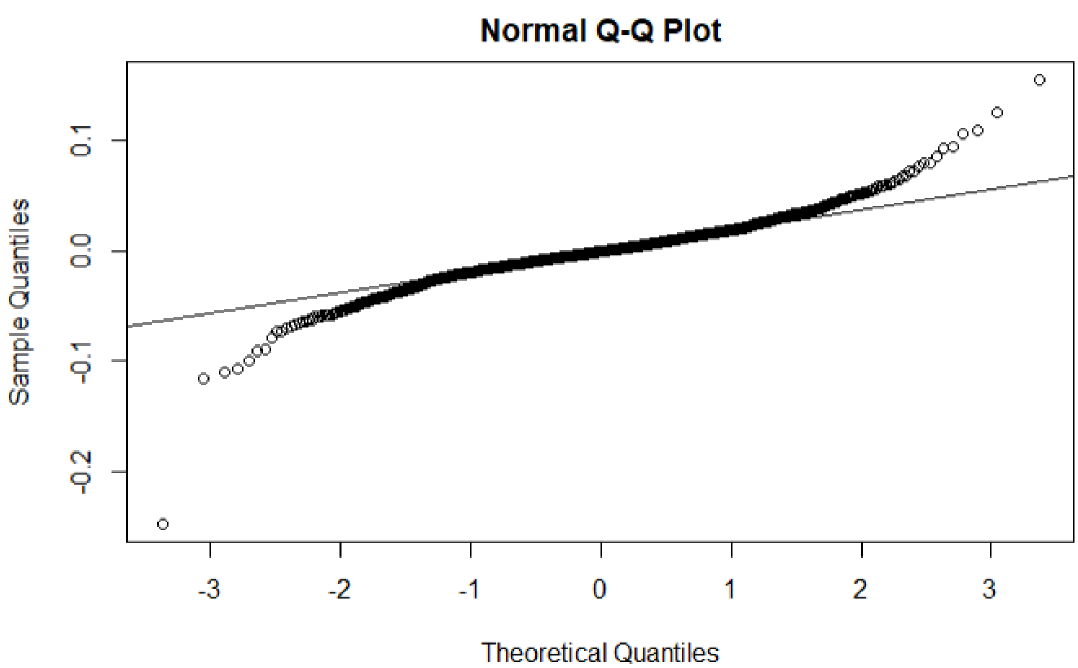
abline(h=0)



Here we see possible reduced variation in the middle of the series and increased variation at the end of the series, not exactly an ideal plot of residuals.

qqnorm(residuals(series2));

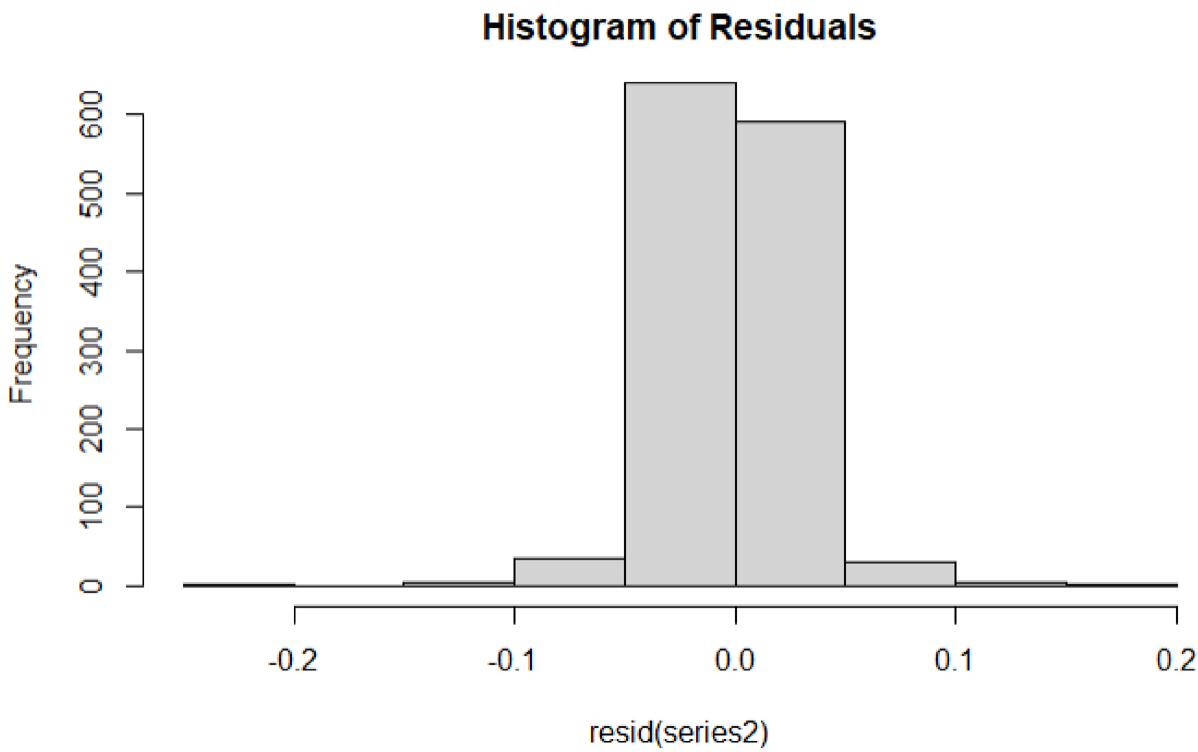
qqline(residuals(series2))



The quantile- quantile plot for the residuals from the ARIMA (1, 1, 1) model for the Netflix stock price time series. Here the extreme values look suspect.

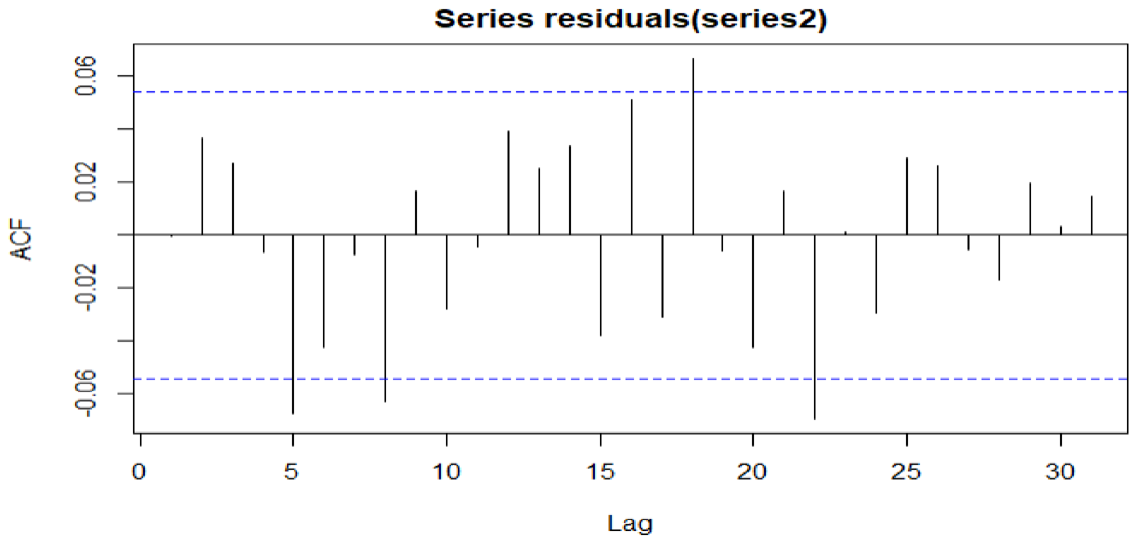
As a statistical test, we can use Shapiro-Wilk test of normality

hist(resid(series2),main="Histogram of Residuals")



This plot demonstrates the histogram of the residuals.

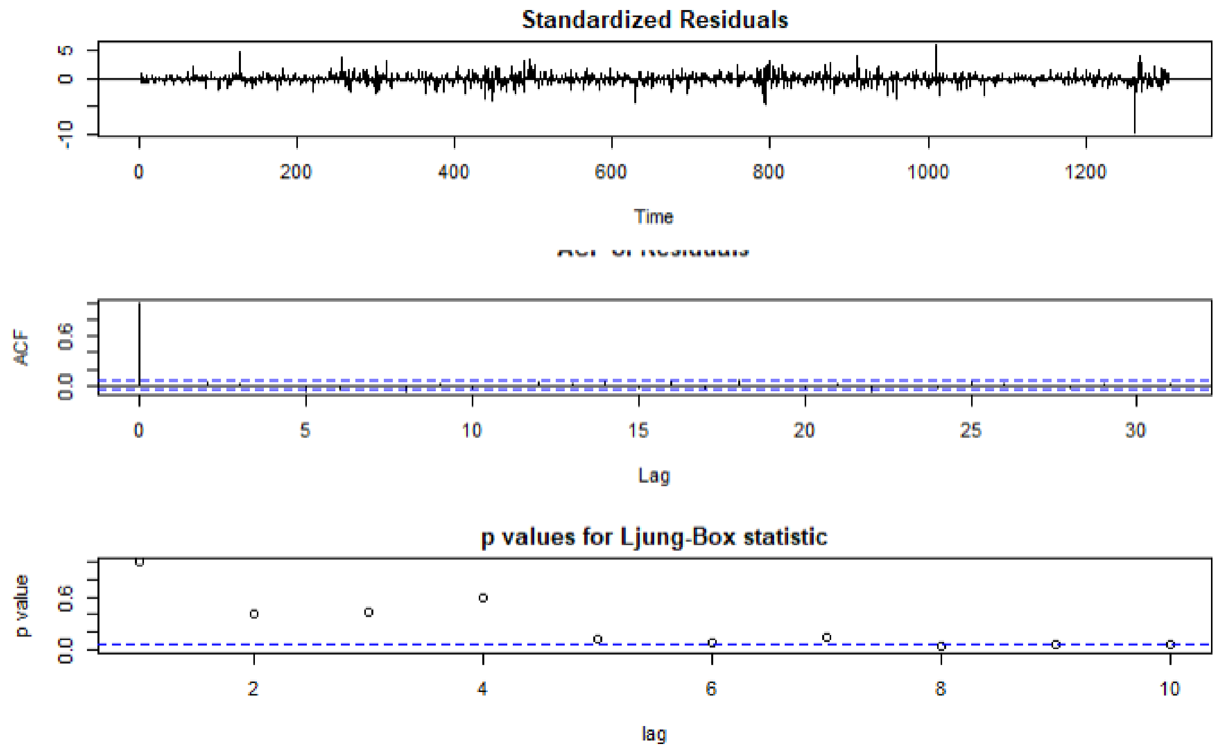
acf(residuals(series2),na.action = na.omit)



Here, we plot the sample ACF of the residuals. There are two strong “statistically significant” correlation here.

Now, we can use Ljung-Box test for this model.

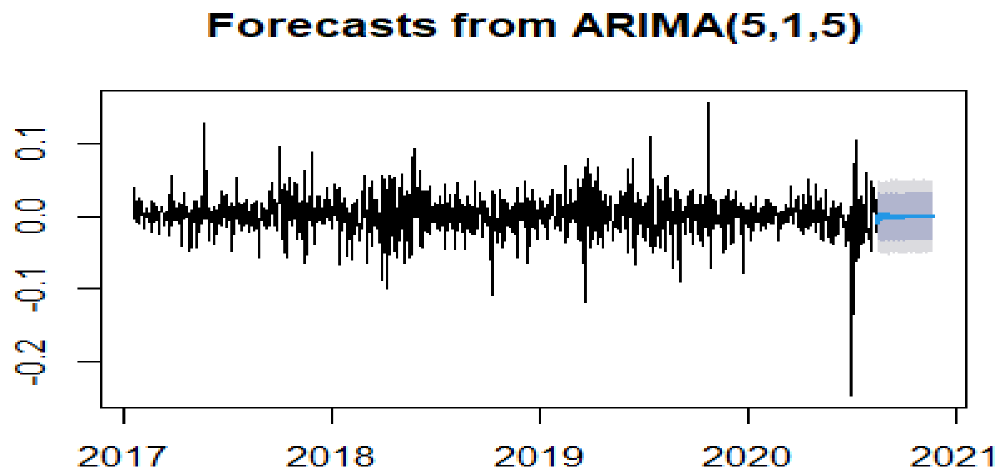
tsdiag(series2)



According to the Ljung-Box test for this model easily we can see a further indication that the model ARIMA (1, 1, 1) shows p value all below 0.05 at lag 6, 8 and 10. It means we reject null hypothesis and our residuals are correlated at some time. As a result, we can say that the ARIMA (5, 1, 5) model has better performance and it evident that the time series is suitable for forecasting.

pred<-forecast(series,h=100)

plot(pred)



Here our forecast for 100 days ahead shows straight line. This is due to nature of arima forecasting tends to be mean reversion. The Ljung Box test shows that the model residuals are non-autocorrelated, suggesting there's no heterocedasticity problem and the model is good, otherwise we might consider GARCH model. The residuals of the model should follow normal distribution and stationary, this is the indication that the arima model fits the data well.

Conclusion

Overall, the forecast yield decent accuracy. The Mean Absolute Error (MAE), or in other words the average magnitude of the errors in a set of predictions (difference between the actual value and the predicted value) is 0.012. The original forecast results in mean reversion (flat line), this is common in arima modeling which means the model cannot capture random events that occurs in a particular period so the forecast tend to follows the mean (mean reverting).