SEASONAL & NON-SEASONAL TIME SERIES ANALYSIS

MA 641

SEASONAL & NON-SEASONAL TIME SERIES ANALYSIS

CO2 LEVEL IN ATMOSPHERE & APPLE STOCK PRICE PREDICTION

OBJECTIVE

The primary objective of this report is to conduct a detailed time series analysis and forecasting for two distinct datasets: atmospheric CO2 levels and Apple Inc. stock prices. This analysis will help in understanding the different behaviors exhibited by seasonal and non-seasonal data and in applying appropriate forecasting models to predict future trends.

DATA OVERVIEW

CO2 Levels in Atmosphere (Seasonal)

Description: Monthly average atmospheric carbon dioxide levels measured at the Mauna Loa Observatory.
Purpose: To examine the seasonal variations and long-term trends in atmospheric CO2 levels, which are crucial for environmental policy planning and climate change mitigation strategies.

Apple Inc. Stock Price (Non-Seasonal)

Description: Daily closing stock prices of Apple Inc., reflecting market behaviors and investor sentiment.
Purpose: To analyze market trends and volatility in Apple's stock price, providing insights for financial investment and stock market behavior.

DATA PREPARATION

Loading Data

The dataset is loaded from a CSV file into R using the read_csv function from the readr package. The initial exploration involves understanding the number of variables, data points, and the basic structure of the dataset.

Data Cleaning

This section describes the process of handling missing values and removing any unneeded variables. It also covers the transformation of the year and month fields into a proper date format, which is essential for time series analysis.

```r
# Load necessary libraries
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(forecast)
```

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

```r
library(tseries)
library(zoo)
```

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

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

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.1     v tidyr   1.3.0
## v stringr 1.5.0
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
##
## Attaching package: 'TSA'
##
## The following object is masked from 'package:readr':
##
##     spec
##
## The following objects are masked from 'package:stats':
##
##     acf, arima
##
## The following object is masked from 'package:utils':
##
##     tar
```

```
# Load the power usage dataset
data <- read_csv("/Users/shivanipatel/Downloads/co2.csv")
```

```
## Rows: 720 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): Year, Month, Decimal Date, Carbon Dioxide (ppm), Seasonally Adjuste...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Convert Year and Month to Date object
data$date <- as.Date(with(data, paste(Year, Month, "01", sep="-")), "%Y-%m-%d")

# Select the relevant columns and remove rows with NAs in 'Carbon Dioxide (ppm)'
data <- data %>%
  select(date, Carbon_Dioxide_ppm = `Carbon Dioxide (ppm)`) %>%
  na.omit()

head(data)
```

```
## # A tibble: 6 x 2
##   date       Carbon_Dioxide_ppm
##   <date>                  <dbl>
## 1 1958-03-01               316.
## 2 1958-04-01               317.
```

```
## 3 1958-05-01              318.
## 4 1958-07-01              316.
## 5 1958-08-01              315.
## 6 1958-09-01              313.
```

```r
# Convert to a time series object
co2_ts <- ts(data$Carbon_Dioxide_ppm, start=c(year(min(data$date)), month(min(data$date))), frequency=1
```

EXPLORATORY DATA ANALYSIS

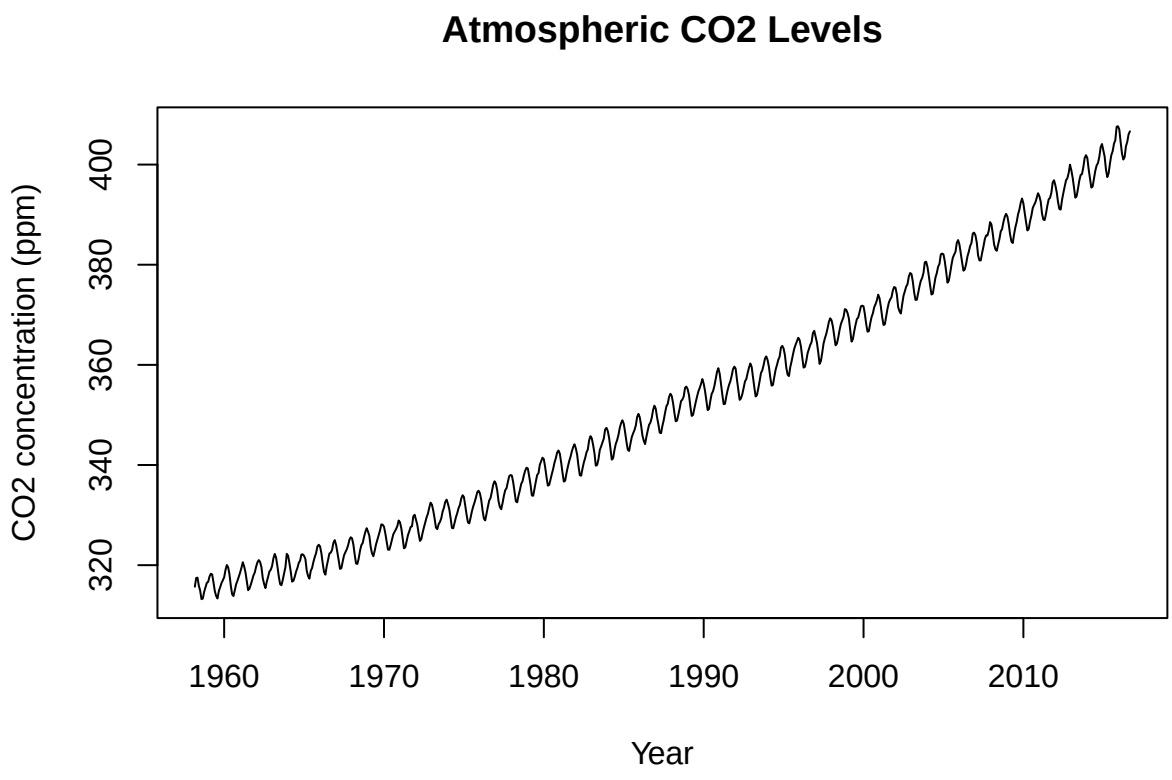
Time Series Plot

A plot of CO2 concentrations over time is created using ggplot2. This visualization helps in identifying any obvious trends, seasonality, or irregular patterns in the data.

Statistical Summary

Provide summary statistics such as mean, median, quartiles, and standard deviation to give a numerical description of the data. This step is crucial for identifying outliers and understanding the distribution of CO2 levels.

```r
# Plot the data
plot(co2_ts, main="Atmospheric CO2 Levels", xlab="Year", ylab="CO2 concentration (ppm)")
```



STATIONARY TESTING

Augmented Dickey-Fuller Test

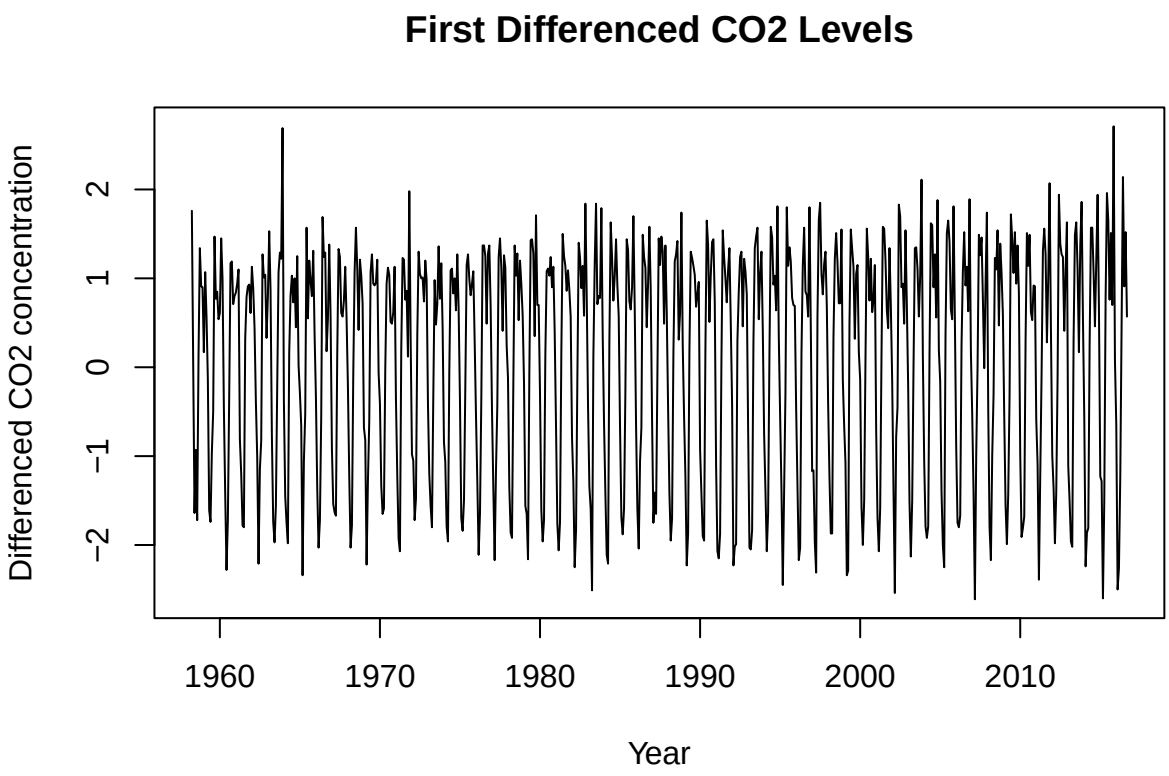
The Augmented Dickey-Fuller (ADF) test is used to test whether the CO2 time series is stationary. This involves explaining the null hypothesis of the ADF test, which assumes the presence of a unit root (non-stationarity).

```
# Perform Augmented Dickey-Fuller Test for Stationarity
adf_test <- adf.test(co2_ts, alternative = "stationary")
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  co2_ts
## Dickey-Fuller = -0.50042, Lag order = 8, p-value = 0.9818
## alternative hypothesis: stationary
```

The Augmented Dickey-Fuller (ADF) test results indicate that the p-value is 0.9818, which is significantly higher than the typical significance level (e.g., 0.05). This suggests that we fail to reject the null hypothesis, implying that the CO2 time series is not stationary.

```
# Differencing the series
co2_diff <- diff(co2_ts, differences = 1)

# Plot the differenced series
plot(co2_diff, main="First Differenced CO2 Levels", xlab="Year", ylab="Differenced CO2 concentration")
```




```
# Perform ADF test on the differenced series
adf_test_diff <- adf.test(co2_diff, alternative = "stationary")
```

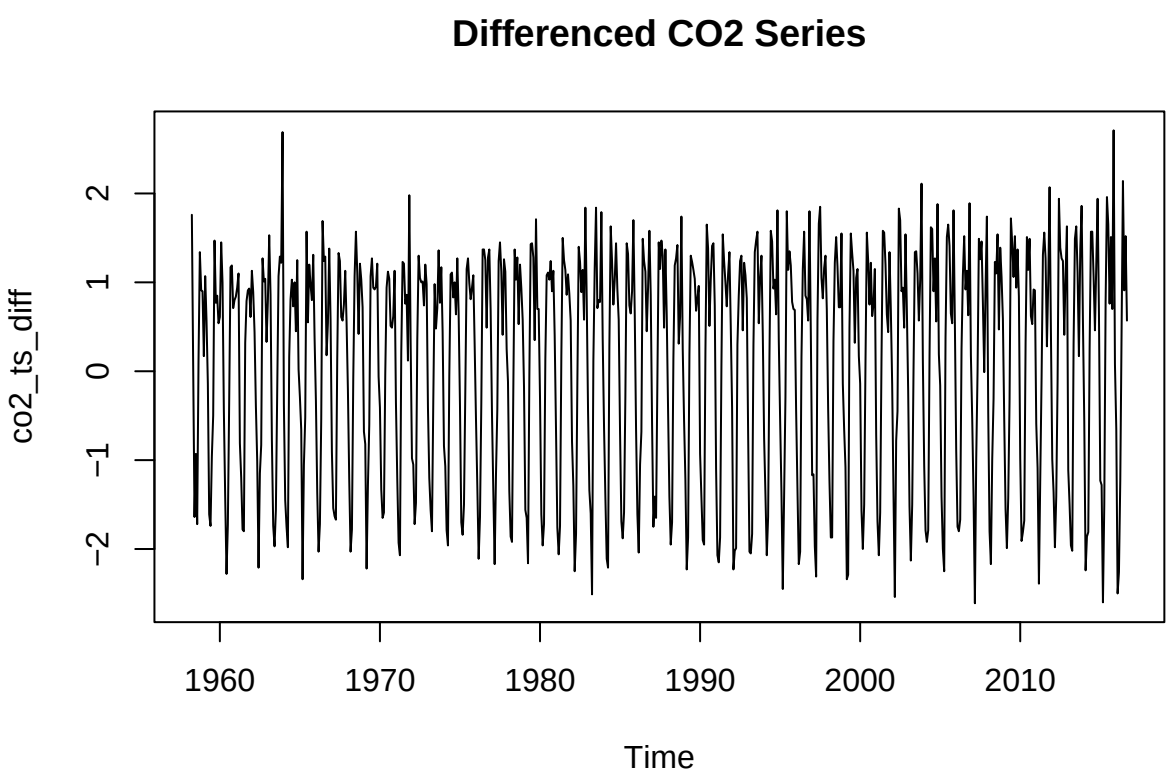
```
## Warning in adf.test(co2_diff, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
print(adf_test_diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  co2_diff
## Dickey-Fuller = -31.713, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
# If the series is not stationary, difference the series
if(adf_test$p.value > 0.05) {
  co2_ts_diff <- diff(co2_ts)
  adf_test_diff <- adf.test(co2_ts_diff, alternative = "stationary")
  plot(co2_ts_diff, main="Differenced CO2 Series")
}
```

```
## Warning in adf.test(co2_ts_diff, alternative = "stationary"): p-value smaller
## than printed p-value
```

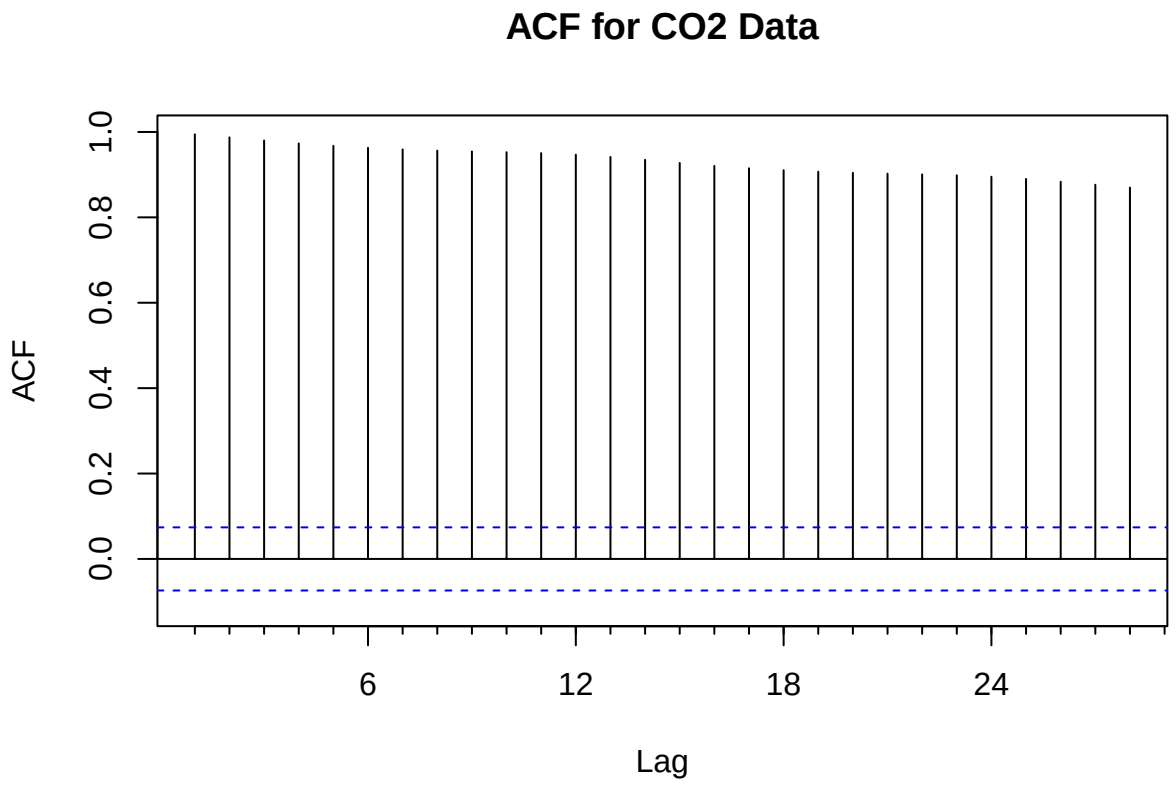


After performing first differencing on the CO2 time series, the Augmented Dickey-Fuller (ADF) test results show a significant improvement towards stationarity. The Dickey-Fuller statistic is -31.713, and despite the warning that the p-value is smaller than the printed value, the reported p-value is 0.01. This indicates that after differencing, the time series has become stationary, as we can reject the null hypothesis of a unit root at the 1% significance level.

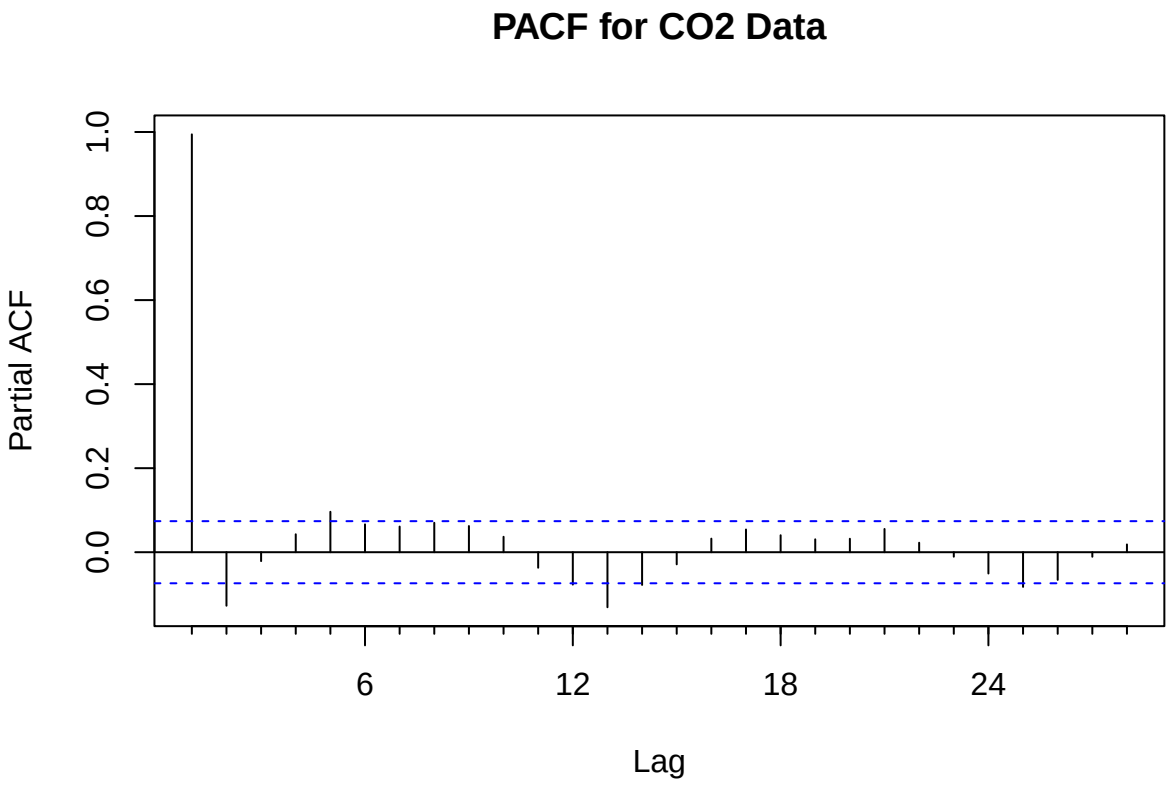
This result means that the first differencing was effective in removing non-stationarity from the time series, likely caused by trends or seasonal components, and it is now suitable for ARIMA or other statistical modeling techniques that assume stationarity.

ACF & PACF PLOTS

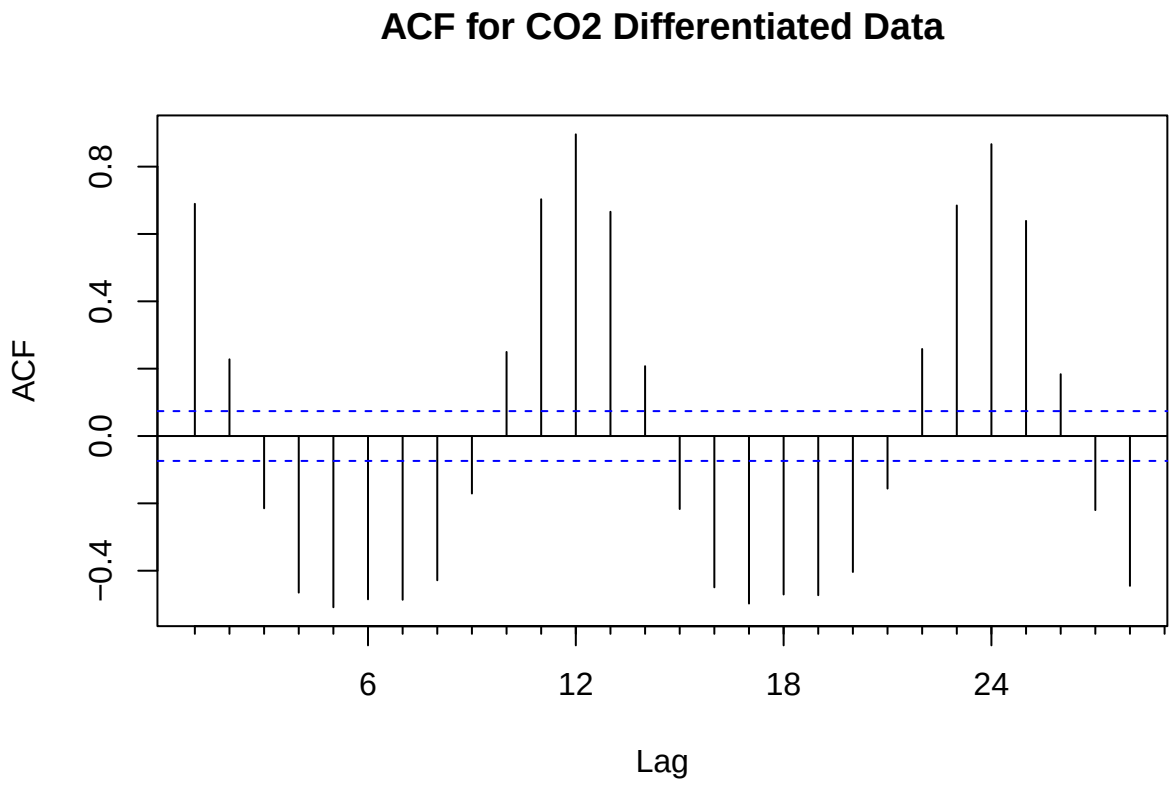
```r
# Autocorrelation plots-Original Data
Acf(co2_ts, main="ACF for CO2 Data")
```



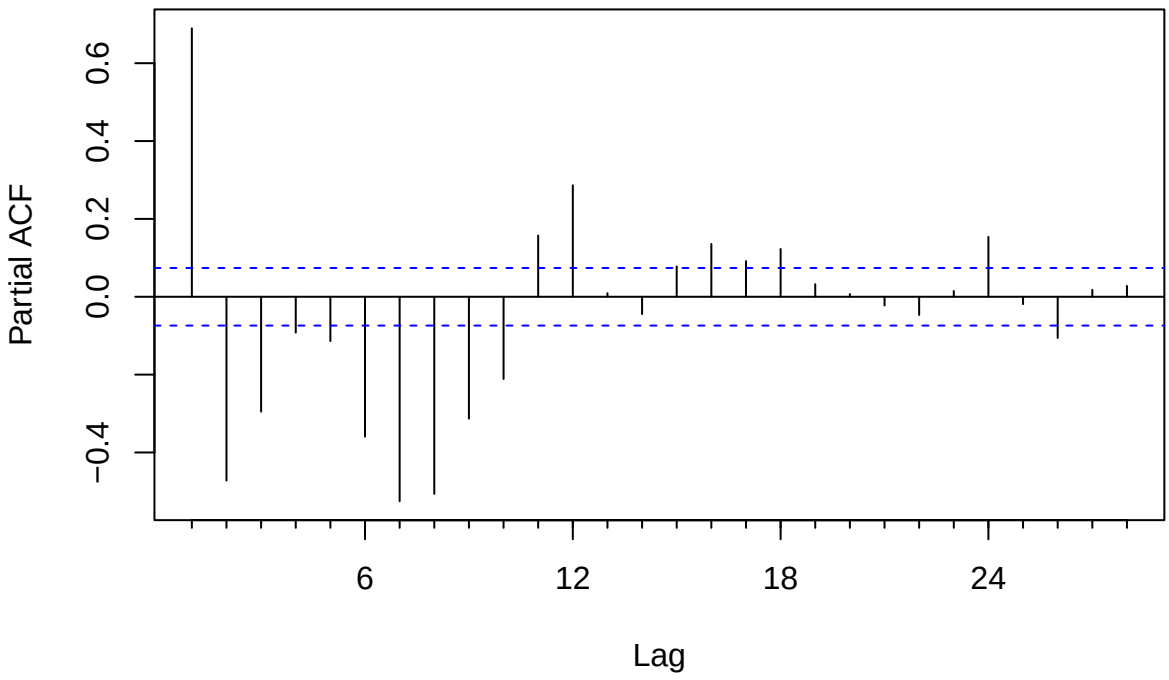

```
# Partial Autocorrelation plots-Original Data
Pacf(co2_ts, main="PACF for CO2 Data")
```




```r
# Autocorrelation plots-Differentiated Data
Acf(co2_ts_diff, main="ACF for CO2 Differentiated Data")
```




```
# Partial Autocorrelation plots-Differentiated Data
Pacf(co2_ts_diff, main="PACF for CO2 Differentiated Data")
```




```
eacf_results <- eacf(co2_ts_diff)
```

```
## 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 x x x x o x x x x x  x  x  x
## 2 x x x x x o o x o x x  x  o  o
## 3 x x o x x o o x o x o  x  x  o
## 4 x o x o o o o x x x o  x  o  o
## 5 x x x o o o o x o x o  x  x  x
## 6 x x x x o o o o x x o  x  o  o
## 7 x x x o o o x o o x o  x  o  o
```

```
# Print the EACF results
print(eacf_results)
```

```
## $eacf
##            [,1]        [,2]       [,3]         [,4]          [,5]         [,6]
## [1,]  0.6895167  0.22775065 -0.2146487 -0.465018026 -0.5084757157 -0.484886993
## [2,]  0.5532924  0.27611857 -0.2647088 -0.333699415 -0.1130799527  0.038089276
## [3,] -0.4813085  0.11698734 -0.1202151 -0.231591193 -0.1712757809 -0.026491692
## [4,] -0.2691305 -0.36111085 -0.0531684 -0.125065579 -0.1244014050  0.043104604
## [5,] -0.4726661  0.03961412 -0.1570199  0.007996732 -0.0032353212  0.006523024
## [6,] -0.3045347 -0.19153712 -0.1299737 -0.012677795  0.0093104166  0.011885156
## [7,] -0.5105202  0.23326114 -0.1295766 -0.081829179 -0.0009944926  0.022745404
```

```
## [8,] -0.5789267  0.26871761 -0.1278199  0.003919743 -0.0188123462  0.055709009
##              [,7]        [,8]         [,9]       [,10]       [,11]      [,12]
## [1,] -0.486265394 -0.42834771 -0.170750442  0.24959706 0.703089828 0.89587519
## [2,] -0.104363441 -0.38796367 -0.154957877  0.14393585 0.448483402 0.71242568
## [3,] -0.022784872 -0.16188357 -0.008152755 -0.08288944 0.089921987 0.29625000
## [4,] -0.041343647 -0.16178513 -0.011767646 -0.10055812 0.040737035 0.24836292
## [5,]  0.039869756 -0.19048665 -0.082800687 -0.09442850 0.017434592 0.19451339
## [6,] -0.001225461 -0.13464649  0.052012241 -0.11770708 0.034975619 0.29086475
## [7,] -0.011607132 -0.02625491  0.084480848 -0.08354389 0.010080741 0.10659741
## [8,]  0.142705181  0.01331788  0.074265791 -0.10057737 0.004763176 0.08371745
##            [,13]       [,14]
## [1,]  0.66590547  0.20739039
## [2,]  0.58637227  0.25707466
## [3,]  0.05233775 -0.05559037
## [4,]  0.30601595 -0.02021271
## [5,] -0.06318851  0.02606753
## [6,] -0.08167440 -0.11594092
## [7,] -0.07357110 -0.01641389
## [8,] -0.06680597  0.01521726
##
## $ar.max
## [1] 8
##
## $ma.ma
## [1] 14
##
## $symbol
##   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" "x" "x" "x" "x" "o" "x" "x" "x" "x" "x" "x" "x" "x"
## 2 "x" "x" "x" "x" "x" "o" "o" "x" "o" "x" "x" "x" "o" "o"
## 3 "x" "x" "o" "x" "x" "o" "o" "x" "o" "x" "o" "x" "x" "o"
## 4 "x" "o" "x" "o" "o" "o" "o" "x" "x" "x" "o" "x" "o" "o"
## 5 "x" "x" "x" "o" "o" "o" "o" "x" "o" "x" "o" "x" "x" "x"
## 6 "x" "x" "x" "x" "o" "o" "o" "o" "x" "x" "o" "x" "o" "o"
## 7 "x" "x" "x" "o" "o" "o" "x" "o" "o" "x" "o" "x" "o" "o"
```

MODELING

ARIMA MODEL

```
arima_model <- Arima(co2_ts_diff, order=c(3, 1, 3))
summary(arima_model)
```

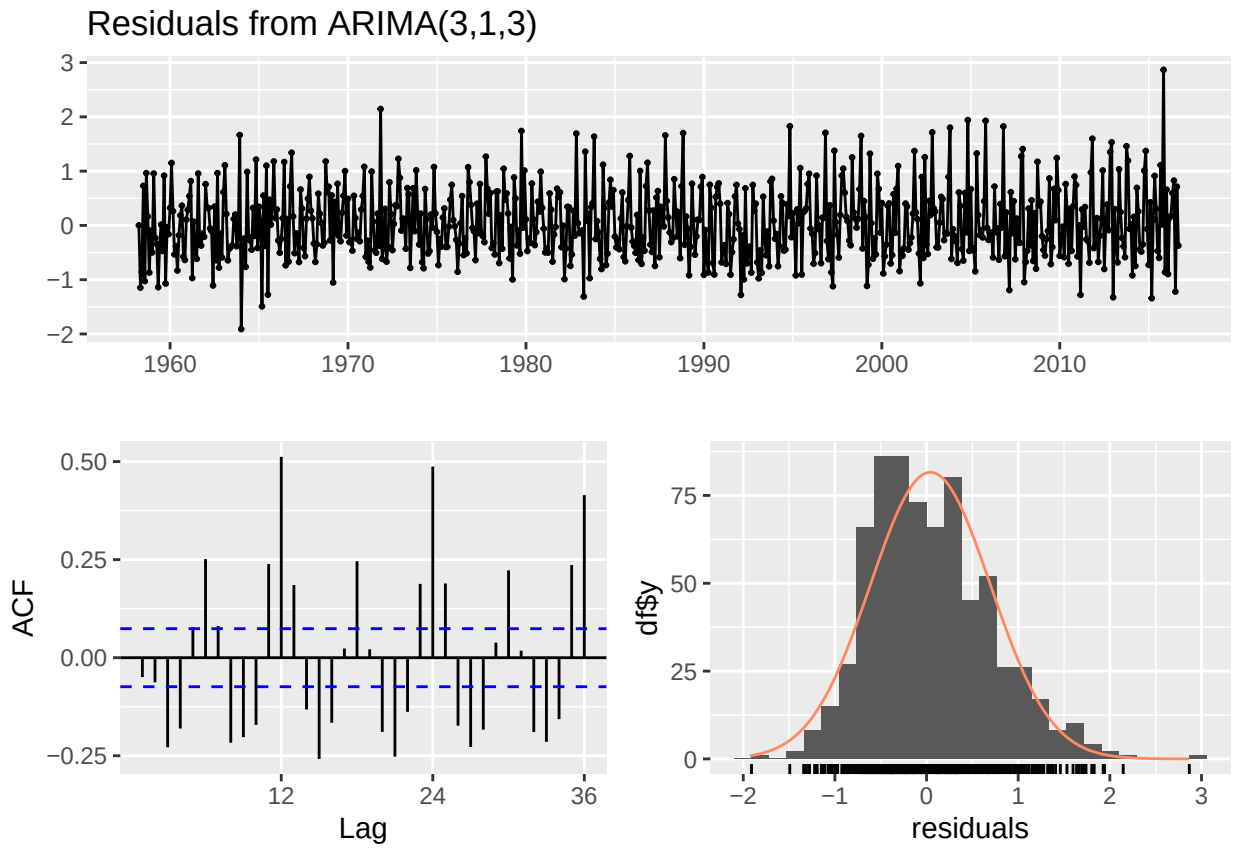
```
## Series: co2_ts_diff
## ARIMA(3,1,3)
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1     ar2      ar3      ma1     ma2     ma3
##       0.5631  0.6391  -0.8159  -0.9631  -0.942  0.9073
## s.e.     NaN     NaN      NaN      NaN     NaN     NaN
##
## sigma^2 = 0.4355:  log likelihood = -705.27
## AIC=1424.54   AICc=1424.7   BIC=1456.41
##
## Training set error measures:
##                       ME      RMSE       MAE  MPE MAPE     MASE        ACF1
## Training set 0.04199303 0.6566576 0.5247205 -Inf  Inf 1.397473 -0.04943005
```

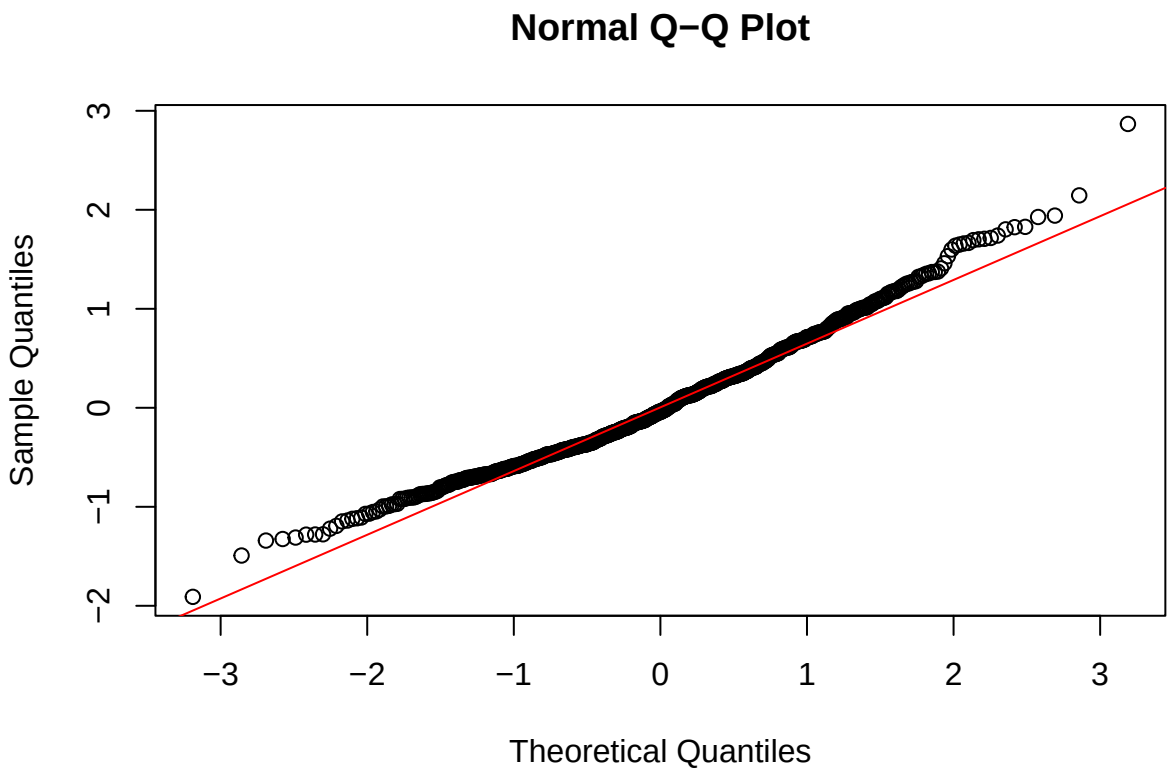
RESIDUAL ANALYSIS

```
# Check the residuals of the ARIMA model
checkresiduals(arima_model)
```



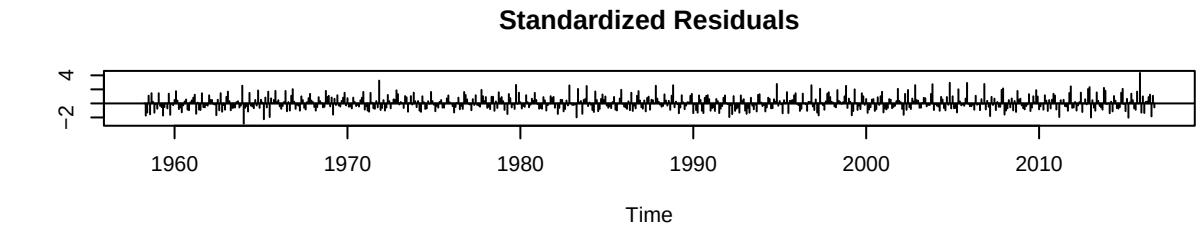

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,3)
## Q* = 864.74, df = 18, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 24
```

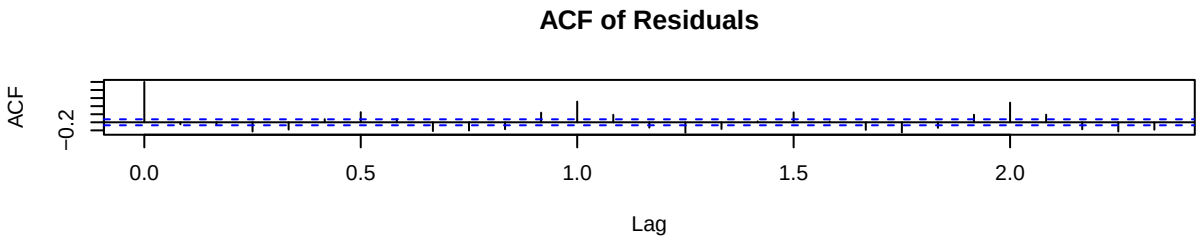
```r
# Assuming you have an ARIMA model stored in a variable called arima_model
residuals <- residuals(arima_model)

qqnorm(residuals)
qqline(residuals, col = "red")
```

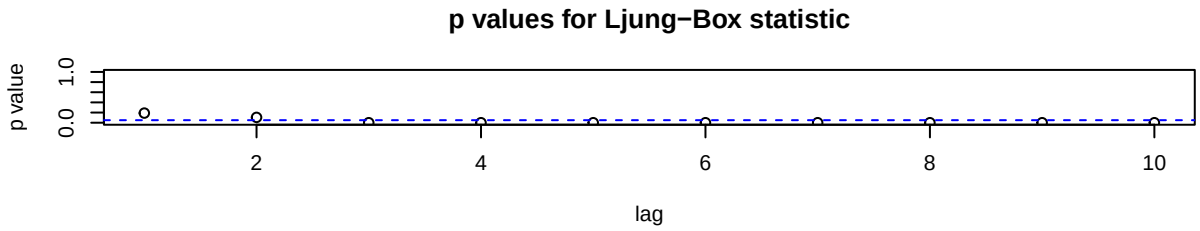


Ljung-Box Test

```r
tsdiag(arima_model)
```

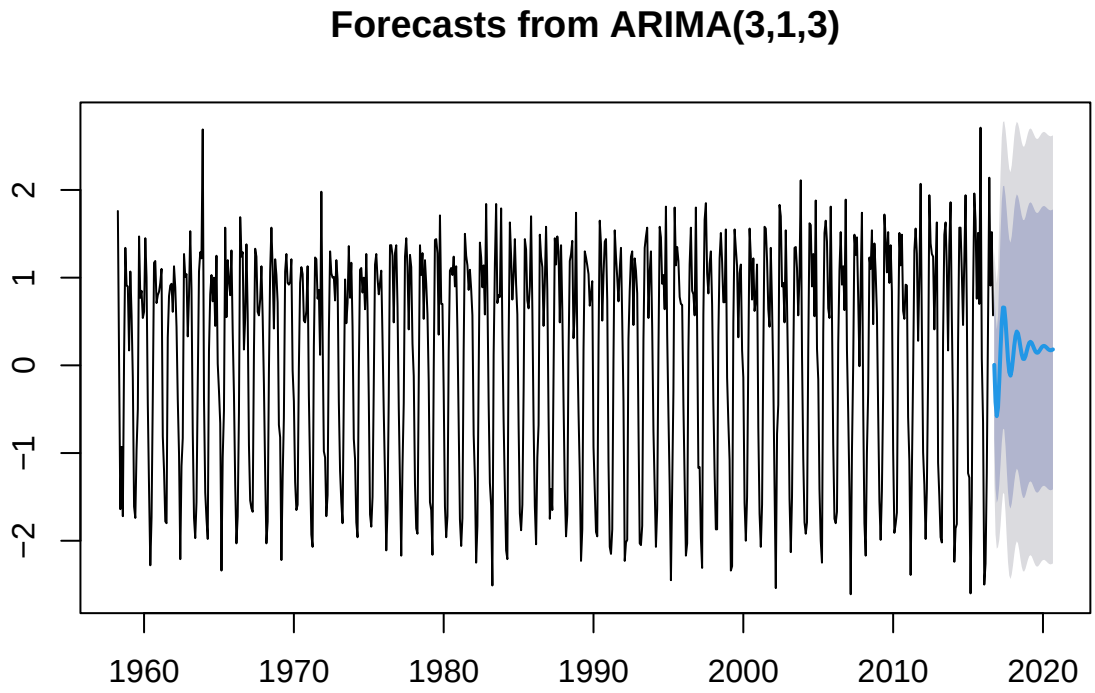




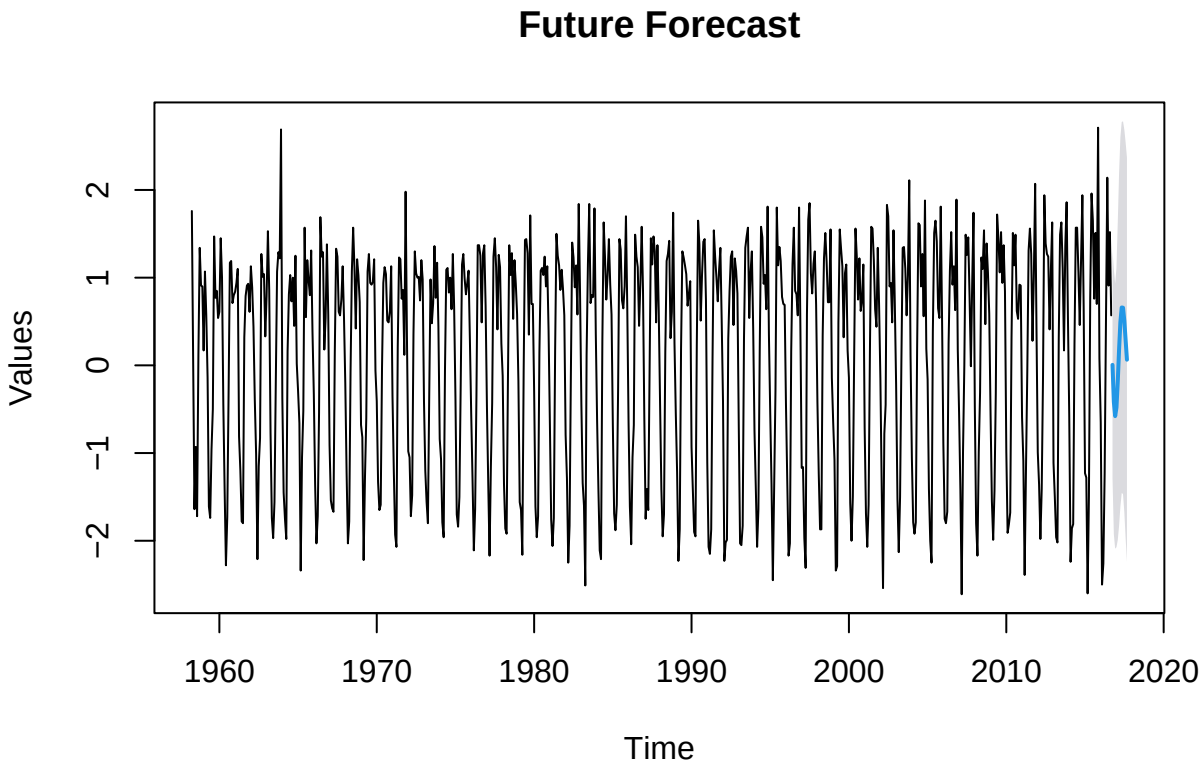


FORECASTING

```
forecasted_values <- forecast(arima_model, h=48)  # forecast the next 48 periods
plot(forecasted_values)
```




```r
# Forecast 12 months into the future
future_forecast <- forecast(arima_model, h=12, level=c(95))  # Default is usually 80 and 95%

# Plot the forecast with the 95% confidence interval
plot(future_forecast, main="Future Forecast", xlab="Time", ylab="Values")
```



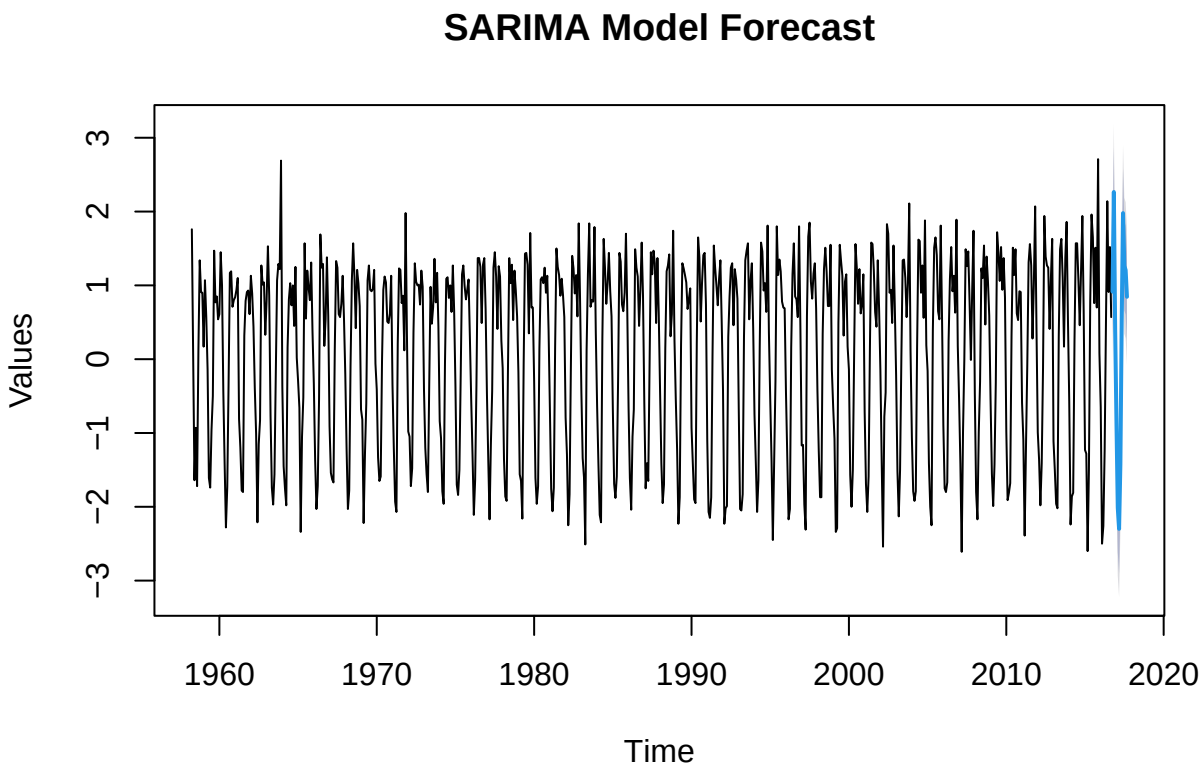
SARIMA MODEL

```
sarima_model <- Arima(co2_ts_diff, order=c(2,1,1), seasonal=list(order=c(1,1,1), period=12))

summary(sarima_model)
```

```
## Series: co2_ts_diff
## ARIMA(2,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1     ar2     ma1     sar1     sma1
##       0.0536  0.0160  -1.000  -0.1252  -0.3756
## s.e.  0.0417  0.0391   0.004   0.1013   0.1024
##
## sigma^2 = 0.2199:  log likelihood = -458.71
## AIC=929.41   AICc=929.54   BIC=956.63
##
## Training set error measures:
##                       ME      RMSE       MAE  MPE MAPE      MASE     ACF1
## Training set 0.02487425 0.4628894 0.3262224 -Inf  Inf 0.8688184 0.008844
```

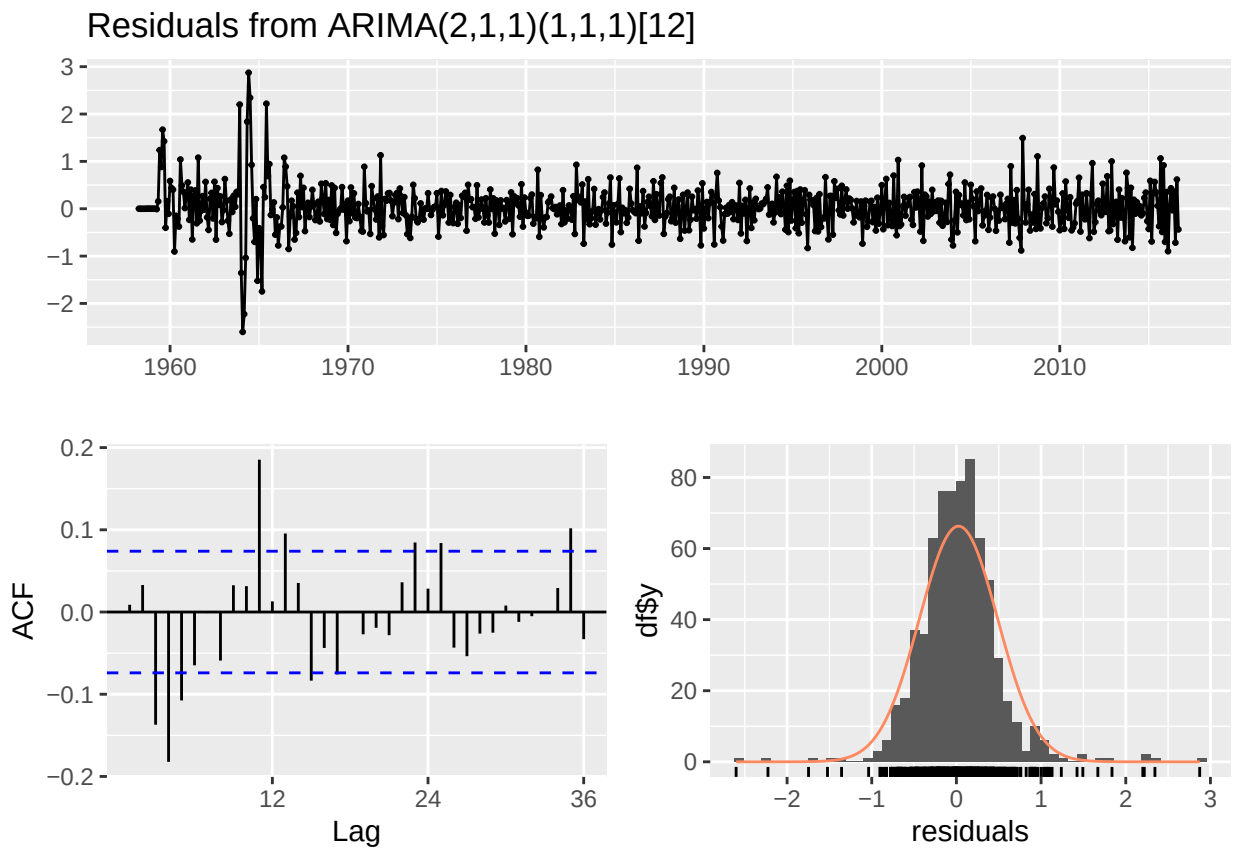
FORECASTING

```
future_forecast_sarima <- forecast(sarima_model, h=12)  # Forecast 12 periods ahead

# Plot the forecast
plot(future_forecast_sarima, main="SARIMA Model Forecast", xlab="Time", ylab="Values")
```



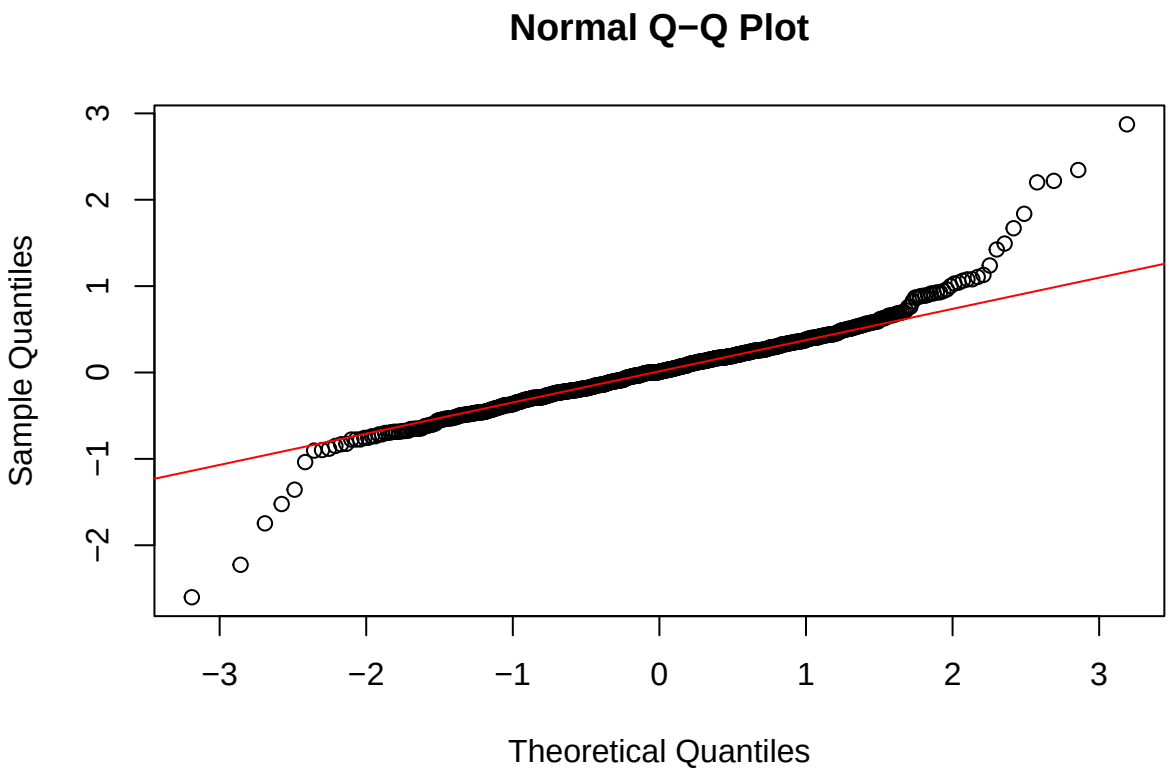
RESIDUAL ANALYSIS

```
checkresiduals(sarima_model)
```



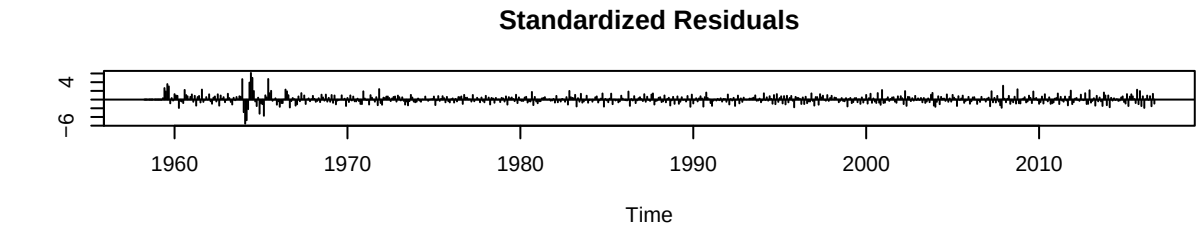

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(1,1,1)[12]
## Q* = 103.5, df = 19, p-value = 1.24e-13
##
## Model df: 5.   Total lags used: 24
```

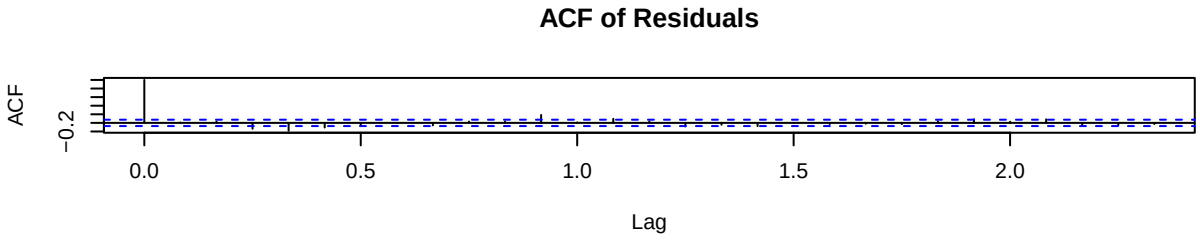
```r
residuals <- residuals(sarima_model)

qqnorm(residuals)
qqline(residuals, col = "red")
```

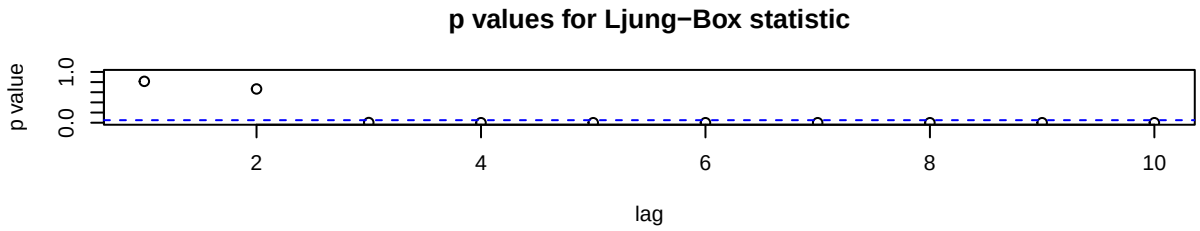


Ljung-Box Test

```
tsdiag(sarima_model)
```

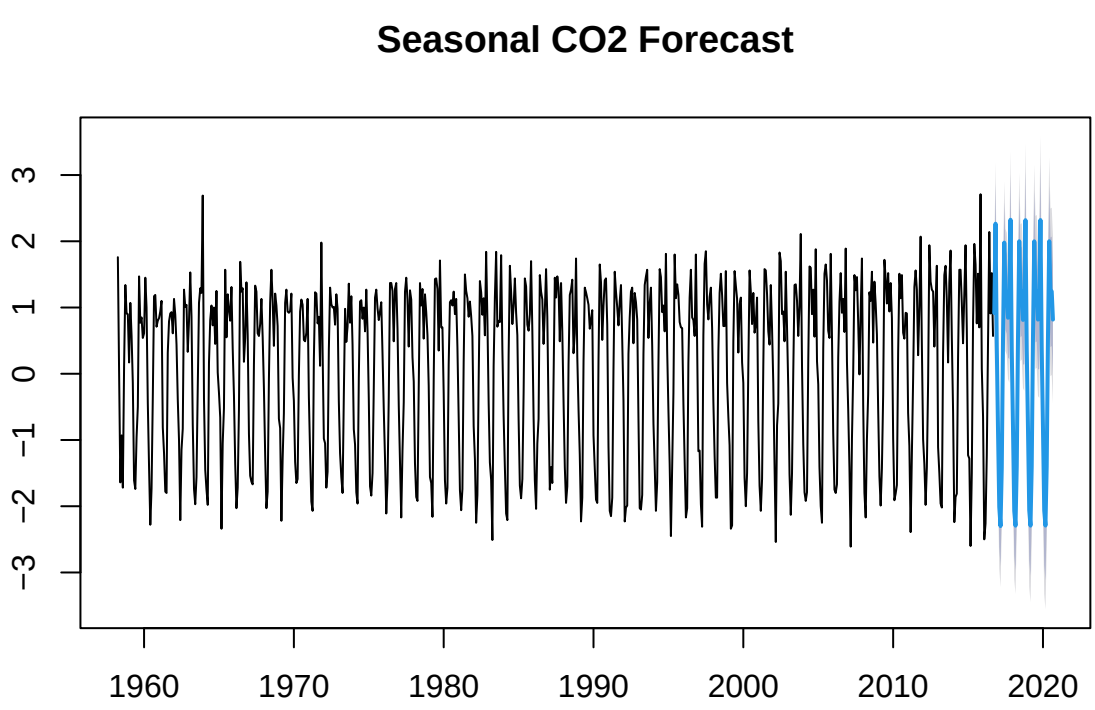






FORECASTING

```
sarima_forecasted_values <- forecast(sarima_model, h=48)
plot(sarima_forecasted_values, main="Seasonal CO2 Forecast")
```




```
sarima_forecasted_values
```

```
##          Point Forecast        Lo 80        Hi 80        Lo 95       Hi 95
## Oct 2016      0.9160933  0.314675691  1.517510829 -0.003695538  1.83588206
## Nov 2016      2.2652772  1.662949590  2.867604731  1.344096634  3.18645769
## Dec 2016      0.4001138 -0.202338277  1.002565791 -0.521257120  1.32148463
## Jan 2017     -0.8464642 -1.448919013 -0.244009287 -1.767839353  0.07491105
## Feb 2017     -2.0117391 -2.614194356 -1.409283765 -2.933114925 -1.09036320
## Mar 2017     -2.2991534 -2.901608724 -1.696698033 -3.220529320 -1.37777744
## Apr 2017     -1.4196142 -2.022069568 -0.817158859 -2.340990168 -0.49823826
## May 2017      0.5313197 -0.071135677  1.133775035 -0.390056277  1.45269564
## Jun 2017      1.9791290  1.376673673  2.581584385  1.057753073  2.90050499
## Jul 2017      1.2656591  0.663203716  1.868114428  0.344283115  2.18703503
## Aug 2017      1.2058320  0.603376660  1.808287372  0.284456059  2.12720797
## Sep 2017      0.8454582  0.243002875  1.447913587 -0.075917726  1.76683419
## Oct 2017      0.8950428  0.221501735  1.568583871 -0.135049368  1.92513497
## Nov 2017      2.3248532  1.651086625  2.998619730  1.294416157  3.35529020
## Dec 2017      0.3613952 -0.312407152  1.035197523 -0.669096563  1.39188693
## Jan 2018     -0.8206784 -1.494482176 -0.146874646 -1.851172343  0.20981552
## Feb 2018     -2.0717285 -2.745532573 -1.397924507 -3.102222882 -1.04123420
## Mar 2018     -2.2931175 -2.966921607 -1.619313475 -3.323611934 -1.26262315
## Apr 2018     -1.3947422 -2.068546285 -0.720938141 -2.425236615 -0.36424781
## May 2018      0.5272793 -0.146524768  1.201083377 -0.503215098  1.55777371
## Jun 2018      2.0004023  1.326598220  2.674206366  0.969907890  3.03089670
## Jul 2018      1.2222643  0.548460246  1.896068392  0.191769916  2.25275872
## Aug 2018      1.2462976  0.572493529  1.920101675  0.215803199  2.27679200
```

```
## Sep 2018      0.8121044  0.138300309  1.485908455 -0.218390021  1.84259879
## Oct 2018      0.8988110  0.144372821  1.653249085 -0.255002602  2.05262451
## Nov 2018      2.3185271  1.563825357  3.073228913  1.164310369  3.47274390
## Dec 2018      0.3673753 -0.387369949  1.122120610 -0.786907965  1.52165863
## Jan 2019     -0.8227740 -1.577521206 -0.068026818 -1.977060236  0.33151221
## Feb 2019     -2.0630853 -2.817832902 -1.308337780 -3.217372126 -0.90879856
## Mar 2019     -2.2927405 -3.047488121 -1.537992907 -3.447027369 -1.13845366
## Apr 2019     -1.3967234 -2.151471032 -0.641975802 -2.551010285 -0.24243655
## May 2019      0.5289178 -0.225829773  1.283665459 -0.625369026  1.68320471
## Jun 2019      1.9988716  1.244124024  2.753619256  0.844584770  3.15315851
## Jul 2019      1.2288299  0.474082292  1.983577524  0.074543038  2.38311678
## Aug 2019      1.2423641  0.487616512  1.997111744  0.088077258  2.39665100
## Sep 2019      0.8174129  0.062665264  1.572160496 -0.336873990  1.97169975
## Oct 2019      0.8994719  0.073647496  1.725296280 -0.363517530  2.16246131
## Nov 2019      2.3204518  1.494381184  3.146522481  1.057085798  3.58381787
## Dec 2019      0.3677593 -0.458354209  1.193872870 -0.895672301  1.63119096
## Jan 2020     -0.8213790 -1.647494614  0.004736708 -2.084813829  0.44205592
## Feb 2020     -2.0630347 -2.889150825 -1.236918669 -3.326470260 -0.79959923
## Mar 2020     -2.2916550 -3.117771151 -1.465538891 -3.555090613 -1.02821943
## Apr 2020     -1.3953427 -2.221458819 -0.569226541 -2.658778287 -0.13190707
## May 2020      0.5298454 -0.296270741  1.355961540 -0.733590210  1.79328101
## Jun 2020      2.0001960  1.174079828  2.826312110  0.736760360  3.26363158
## Jul 2020      1.2291406  0.403024471  2.055256753 -0.034294997  2.49257622
## Aug 2020      1.2439893  0.417873142  2.070105424 -0.019446326  2.50742489
## Sep 2020      0.8178810 -0.008235173  1.643997109 -0.445554641  2.08131658
```

CONCLUSION

The comparison of SARIMA and ARIMA models for forecasting CO2 emissions reveals their respective capacities to handle seasonal and non-seasonal data. The SARIMA model, specifically tailored for seasonal data, effectively captures the periodic fluctuations in CO2 levels, as demonstrated by the consistent patterns over the decades and the highlighted forecast extending these trends. Conversely, the ARIMA model, though applied to the same dataset, does not inherently account for seasonal variations, which may result in less accuracy for long-term predictions where seasonal effects are pronounced. The choice between these models should consider the data's seasonal characteristics and the forecasting goals, with SARIMA being preferable for capturing and projecting cyclical behaviors inherent in atmospheric CO2 measurements.

NON-SEASONAL DATA

```
library(stats)
library(tseries)
library(tidyverse)
library(TSA)
library(ggplot2)
library(rugarch)
```

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

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

```
## The following object is masked from 'package:purrr':
##
##     reduce

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

```
library(forecast)
library(dplyr)
```

```
data1 <- read.csv("/Users/shivanipatel/Downloads/AAPL.csv")
head(data1)
```

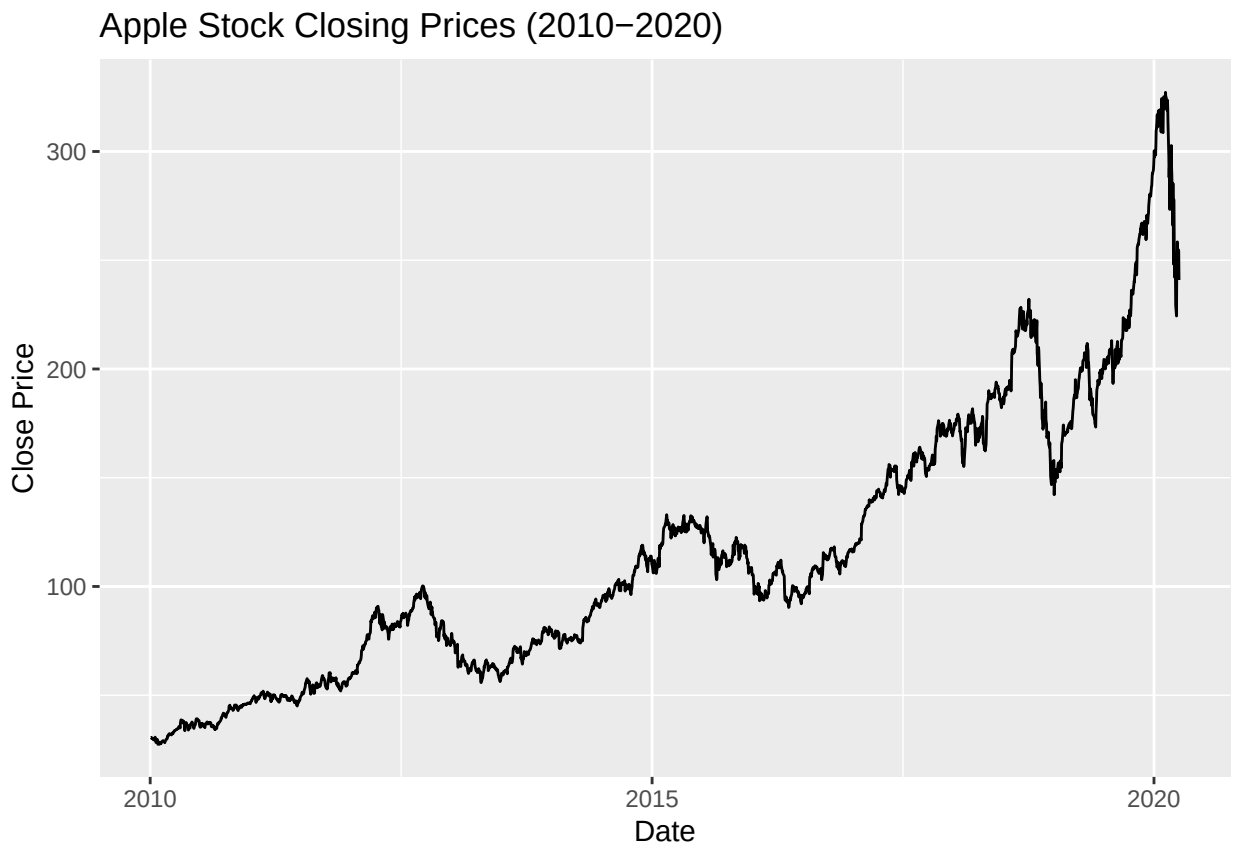
```
##         Date      Open      High       Low     Close Adj.Close    Volume
## 1 1980-12-12 0.5133929 0.5156250 0.5133929 0.5133929 0.4067816 117258400
## 2 1980-12-15 0.4888393 0.4888393 0.4866071 0.4866071 0.3855582  43971200
## 3 1980-12-16 0.4531250 0.4531250 0.4508929 0.4508929 0.3572603  26432000
## 4 1980-12-17 0.4620536 0.4642857 0.4620536 0.4620536 0.3661034  21610400
## 5 1980-12-18 0.4754464 0.4776786 0.4754464 0.4754464 0.3767152  18362400
## 6 1980-12-19 0.5044643 0.5066964 0.5044643 0.5044643 0.3997071  12157600
```

```
data<- data1 %>%
    filter(year(Date) >= 2010 & year(Date) <= 2020)

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

```
summary(data)
```

```
##       Date                 Open             High             Low
##  Min.   :2010-01-04   Min.   : 27.48   Min.   : 28.00   Min.   : 27.18
##  1st Qu.:2012-07-24   1st Qu.: 65.48   1st Qu.: 66.02   1st Qu.: 64.84
##  Median :2015-02-18   Median :100.57   Median :101.54   Median : 99.91
##  Mean   :2015-02-15   Mean   :114.77   Mean   :115.86   Mean   :113.69
##  3rd Qu.:2017-09-07   3rd Qu.:155.37   3rd Qu.:156.69   3rd Qu.:153.96
##  Max.   :2020-04-01   Max.   :324.74   Max.   :327.85   Max.   :323.35
##      Close          Adj.Close          Volume
##  Min.   : 27.44   Min.   : 23.82   Min.   : 11362000
##  1st Qu.: 65.37   1st Qu.: 57.66   1st Qu.: 31117400
##  Median :100.80   Median : 93.21   Median : 54777800
##  Mean   :114.81   Mean   :108.44   Mean   : 74271683
##  3rd Qu.:155.42   3rd Qu.:150.39   3rd Qu.:100917550
##  Max.   :327.20   Max.   :327.20   Max.   :470249500
```

```
# Plotting the closing prices over time
ggplot(data, aes(x = Date, y = Close)) +
  geom_line() +
  labs(title = "Apple Stock Closing Prices (2010-2020)", x = "Date", y = "Close Price")
```



The chart of Apple's stock prices from 2010 to 2020 suggests the data is non-stationary. This is indicated by the overall upward trend in the stock prices and the significant fluctuations, including sharp increases and sudden declines.

```
sum(is.na(data$Close))
```

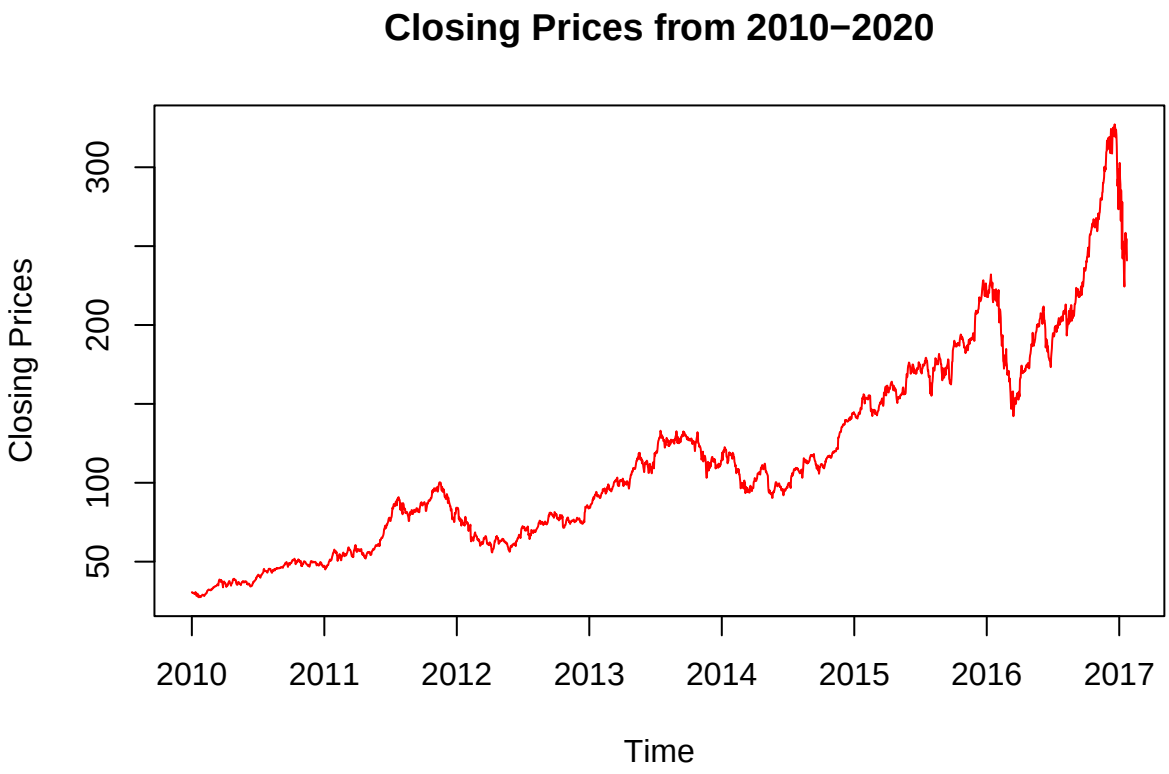
```
## [1] 0
```

```
data <- select(data, Date, Close)
```

```
apple_ts <- ts(data=data$Close, start = c(2010, 1), frequency = 365.25)

summary(apple_ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.44   65.37  100.80  114.81  155.42  327.20
```

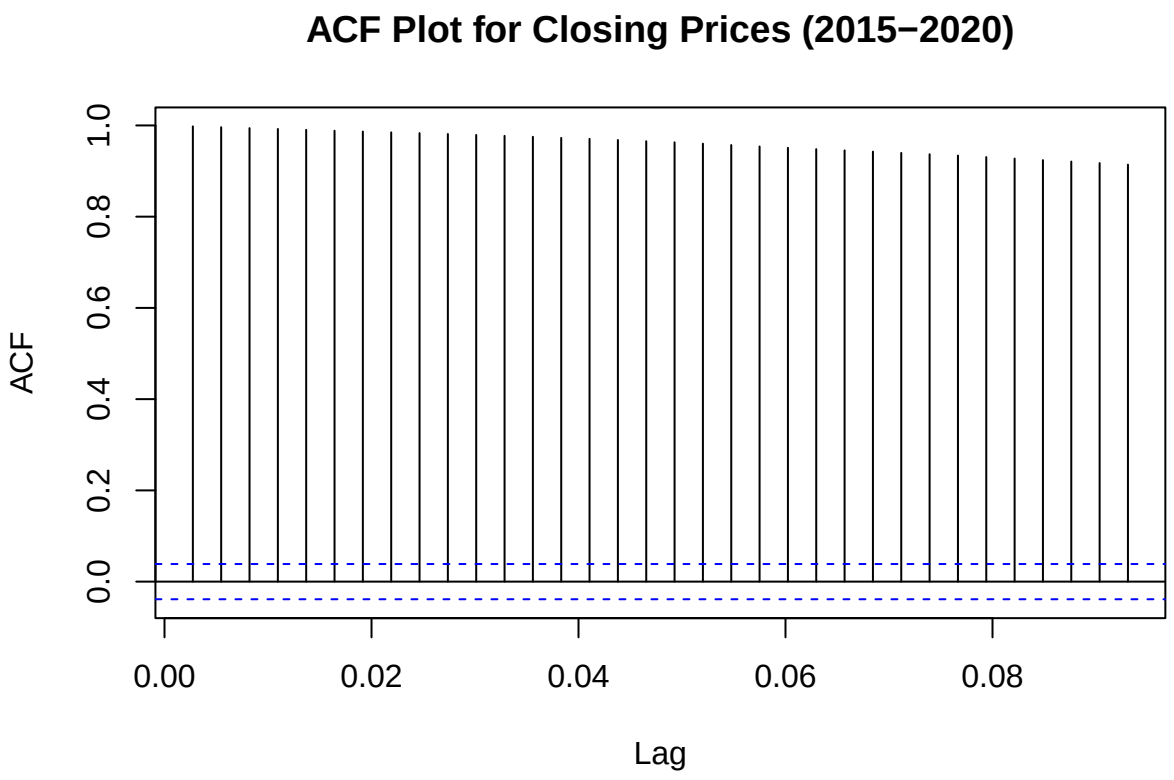
```
plot(apple_ts, type='l', col='red', ylab="Closing Prices", main="Closing Prices from 2010-2020")
```



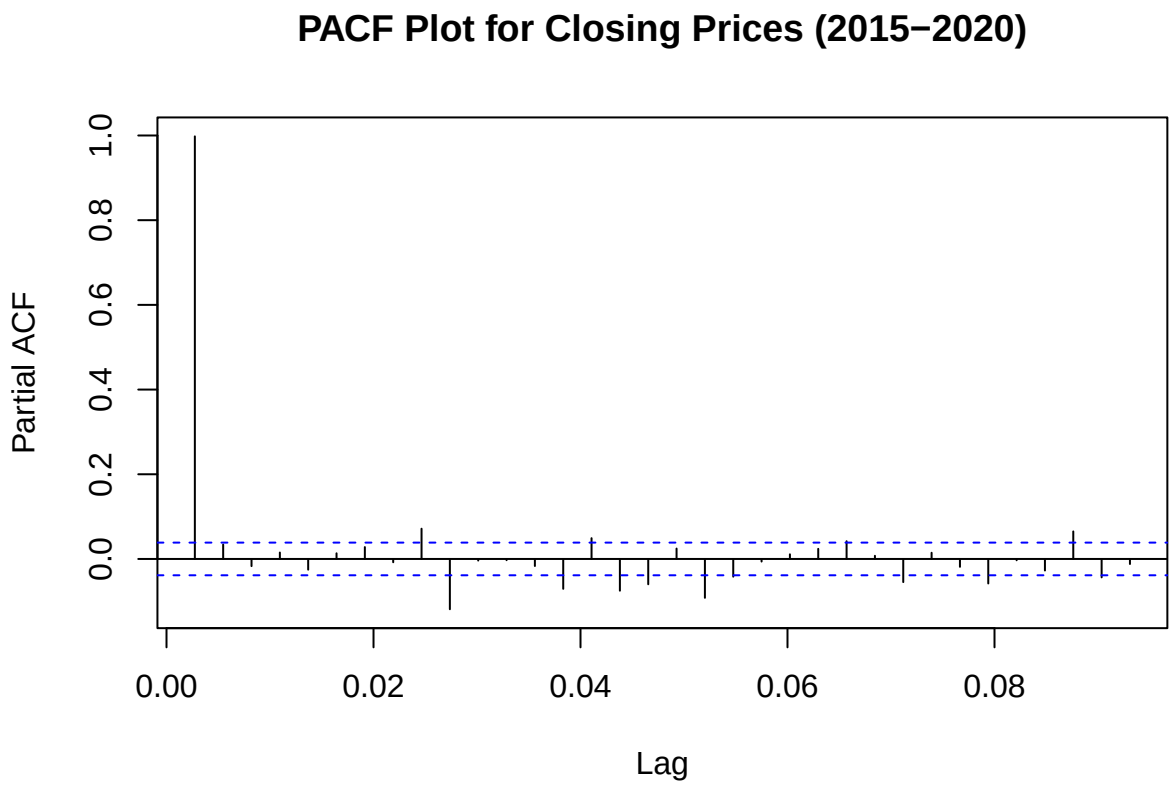
The plot of Apple's closing prices from 2010 to 2020 shows a clear upward trend with significant fluctuations, indicating non-stationarity in the time series data.

ACF & PACF PLOTS

```
acf(apple_ts, main="ACF Plot for Closing Prices (2015-2020)")
```




```
pacf(apple_ts, main="PACF Plot for Closing Prices (2015-2020)")
```




```
eacf(apple_ts)
```

```
## 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 x x x x x o o  x  x  x
## 2 o x o o o o x x x x o  o  o  x
## 3 x x o o o o x o x o o  o  o  o
## 4 x x o o o o o x o x o  o  o  o
## 5 x x x o o o o x x x o  o  o  o
## 6 x x x o o x o o o o x  o  o  o
## 7 x x x o x x o o x o x  o  o  o
```

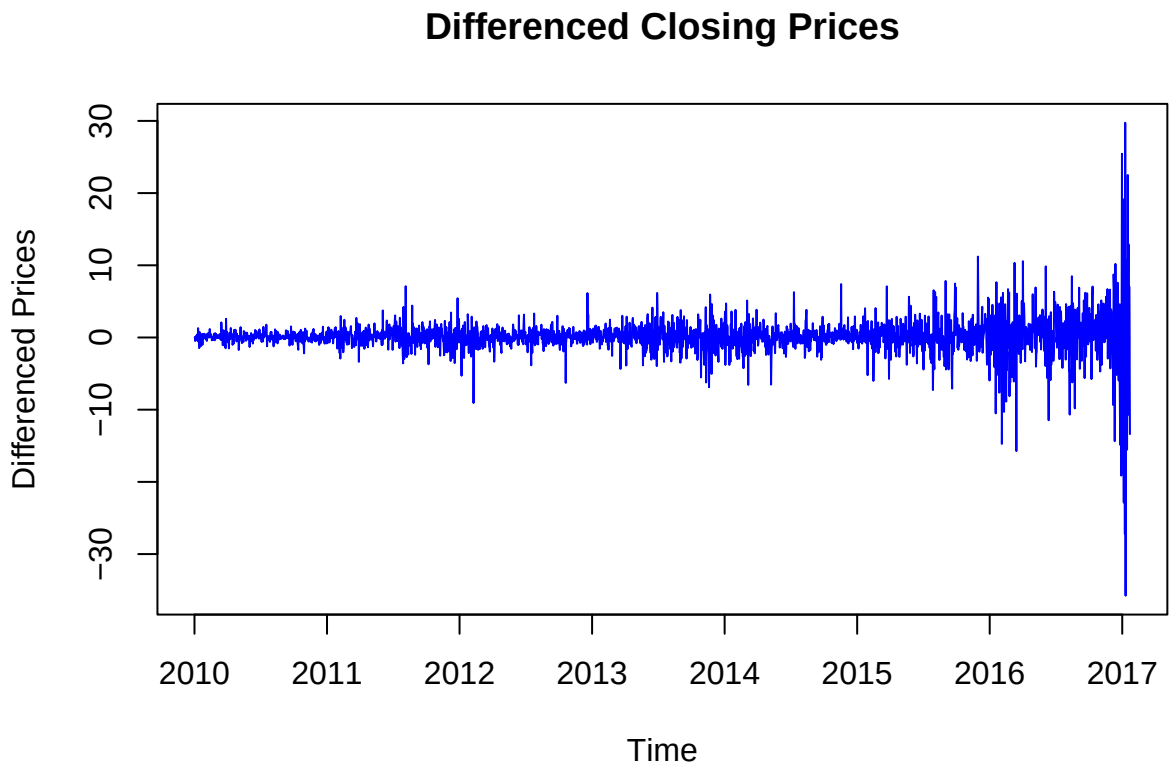
Augmented Dickey-Fuller Test

```
adf_result = adf.test(apple_ts, alternative = "stationary")
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  apple_ts
## Dickey-Fuller = -2.4465, Lag order = 13, p-value = 0.3893
## alternative hypothesis: stationary
```

The Augmented Dickey-Fuller test result with a p-value of 0.3893 indicates that the Apple stock price series is non-stationary, as we cannot reject the null hypothesis of a unit root.

```
diff_apple_ts <- diff(apple_ts)

# Plotting the differenced time series
plot(diff_apple_ts, type='l', col='blue', main="Differenced Closing Prices", ylab="Differenced Prices")
```



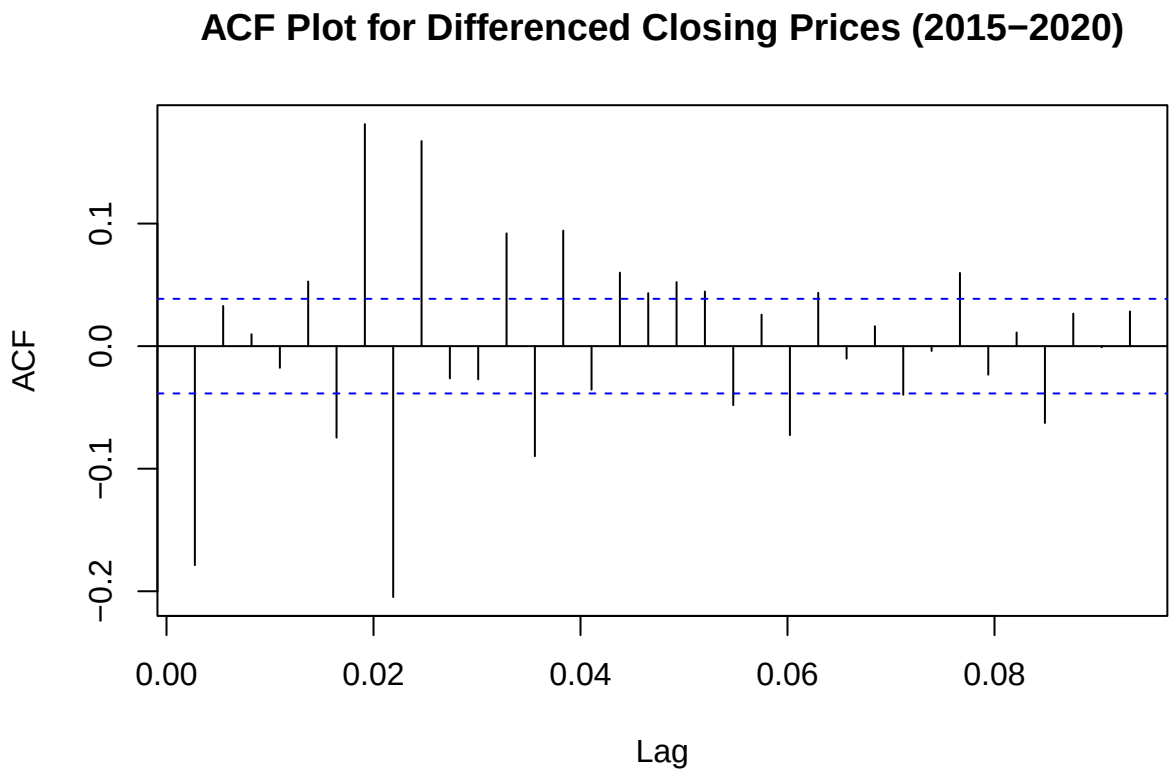

```
adf_result_diff = adf.test(diff_apple_ts, alternative = "stationary")
```

```
## Warning in adf.test(diff_apple_ts, alternative = "stationary"): p-value smaller
## than printed p-value
```

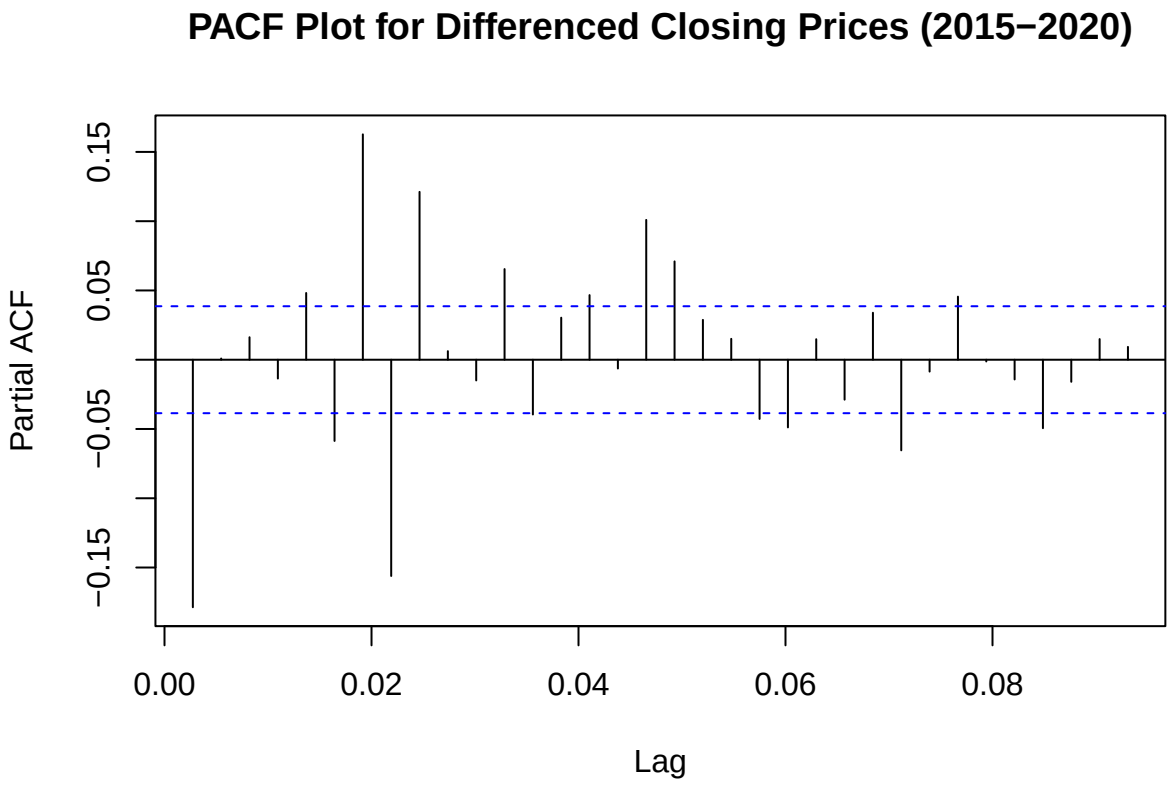
```
print(adf_result_diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_apple_ts
## Dickey-Fuller = -12.001, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf(diff_apple_ts, main="ACF Plot for Differenced Closing Prices (2015-2020)")
```




```r
pacf(diff_apple_ts, main="PACF Plot for Differenced Closing Prices (2015-2020)")
```




```
eacf(diff_apple_ts)
```

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

Based on the eacf, we could consider ARIMA(1,1,0), ARIMA(2,1,0), ARIMA(1,1,1), ARIMA(2,1,1), ARIMA(3,1,2).

```
ARIMA110 = Arima(apple_ts, order = c(1,1,0), method="ML")
ARIMA110
```

```
## Series: apple_ts
## ARIMA(1,1,0)
##
## Coefficients:
##           ar1
##       -0.1792
```

```
## s.e.   0.0195
##
## sigma^2 = 6.56:  log likelihood = -6082.18
## AIC=12168.37   AICc=12168.37   BIC=12180.08
```

```
ARIMA210 = Arima(apple_ts, order = c(2,1,0), method="ML")
ARIMA210
```

```
## Series: apple_ts
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1     ar2
##       -0.1789  0.0020
## s.e.   0.0198  0.0198
##
## sigma^2 = 6.563:  log likelihood = -6082.18
## AIC=12170.36   AICc=12170.36   BIC=12187.92
```

```
ARIMA111 = Arima(apple_ts, order = c(1,1,1), method="ML")
ARIMA111
```

```
## Series: apple_ts
## ARIMA(1,1,1)
##
## Coefficients:
##           ar1     ma1
##       -0.1881  0.0090
## s.e.   0.0987  0.1002
##
## sigma^2 = 6.563:  log likelihood = -6082.18
## AIC=12170.36   AICc=12170.37   BIC=12187.92
```

```
ARIMA211 = Arima(apple_ts, order = c(2,1,1), method="ML")
ARIMA211
```

```
## Series: apple_ts
## ARIMA(2,1,1)
##
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs produced

##           ar1     ar2      ma1
##       -0.0884  0.0198  -0.0902
## s.e.      NaN     NaN      NaN
##
## sigma^2 = 6.565:  log likelihood = -6082.17
## AIC=12172.33   AICc=12172.35   BIC=12195.75
```

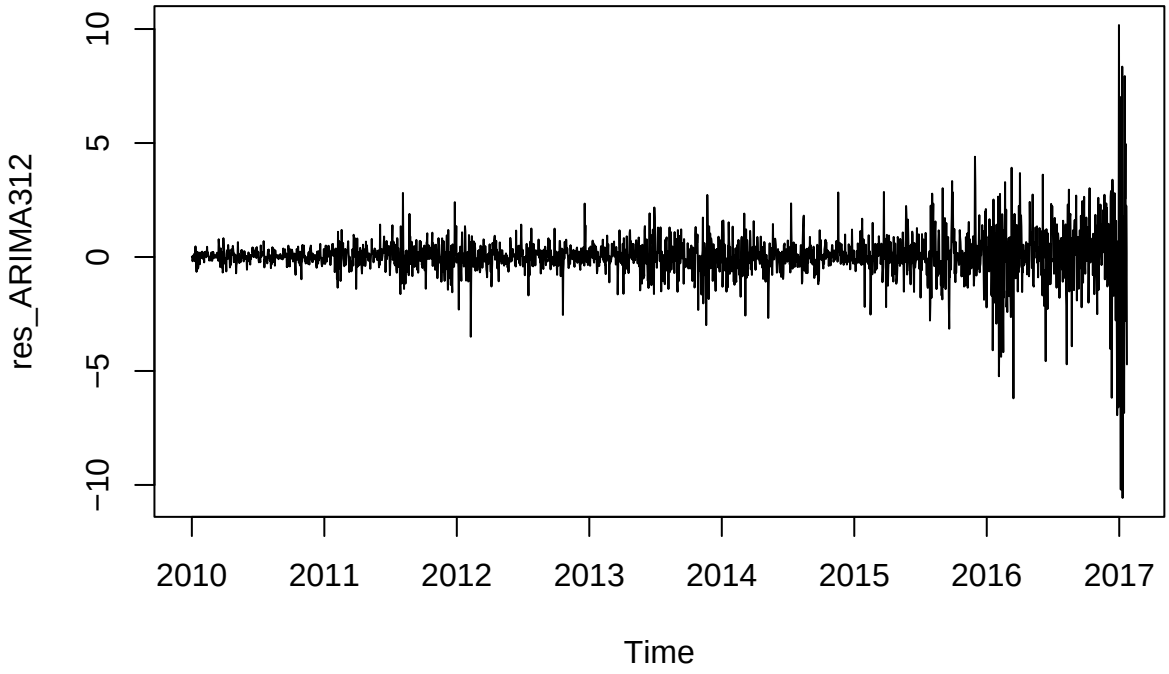
```
ARIMA312 = Arima(apple_ts, order = c(3,1,2), method="ML")
ARIMA312
```

```
## Series: apple_ts
## ARIMA(3,1,2)
##
## Coefficients:
##           ar1     ar2     ar3      ma1      ma2
##       -0.1622  0.8115  0.2299  -0.0103  -0.8434
## s.e.   0.0358  0.0322  0.0194   0.0317   0.0302
##
## sigma^2 = 6.416:  log likelihood = -6051.6
## AIC=12115.21   AICc=12115.24   BIC=12150.33
```

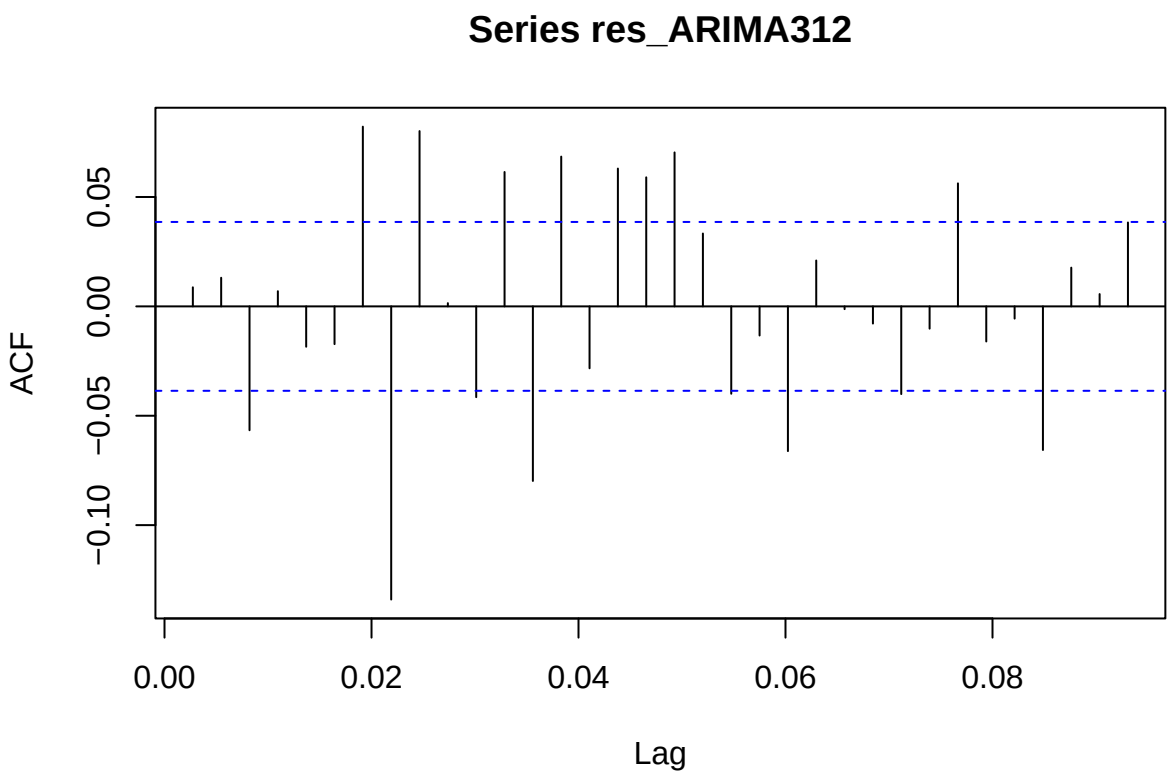
ARIMA312 performs the best with the lowest aic value.

RESIDUAL ANALYSIS

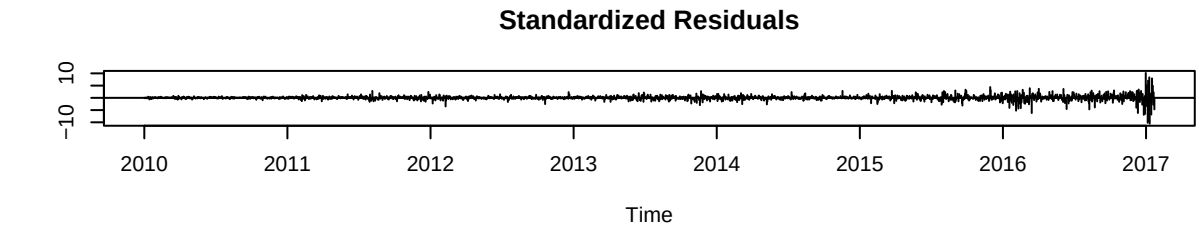
```
res_ARIMA312 = rstandard(ARIMA312)
plot(res_ARIMA312)
```

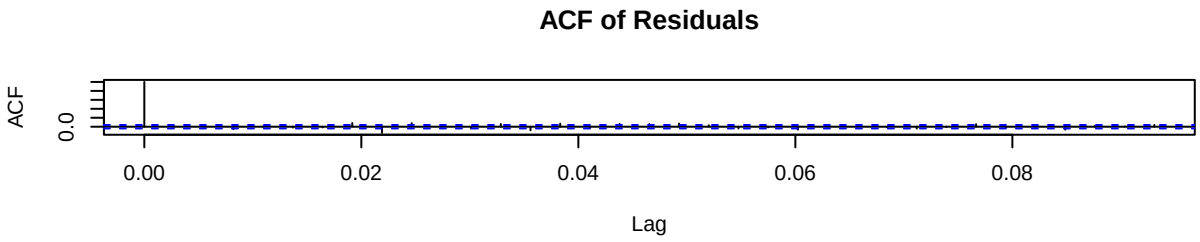


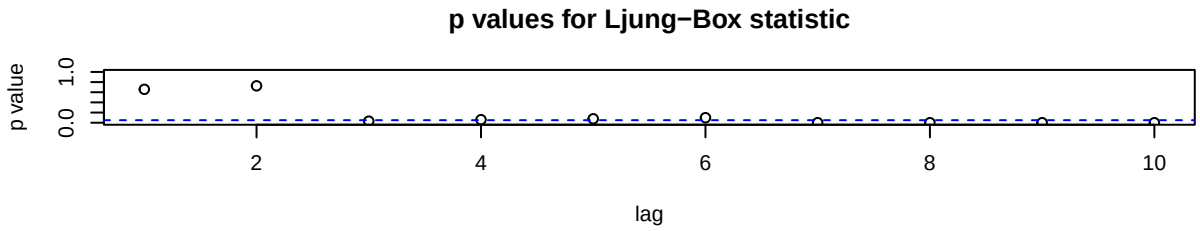
```
acf(res_ARIMA312)
```



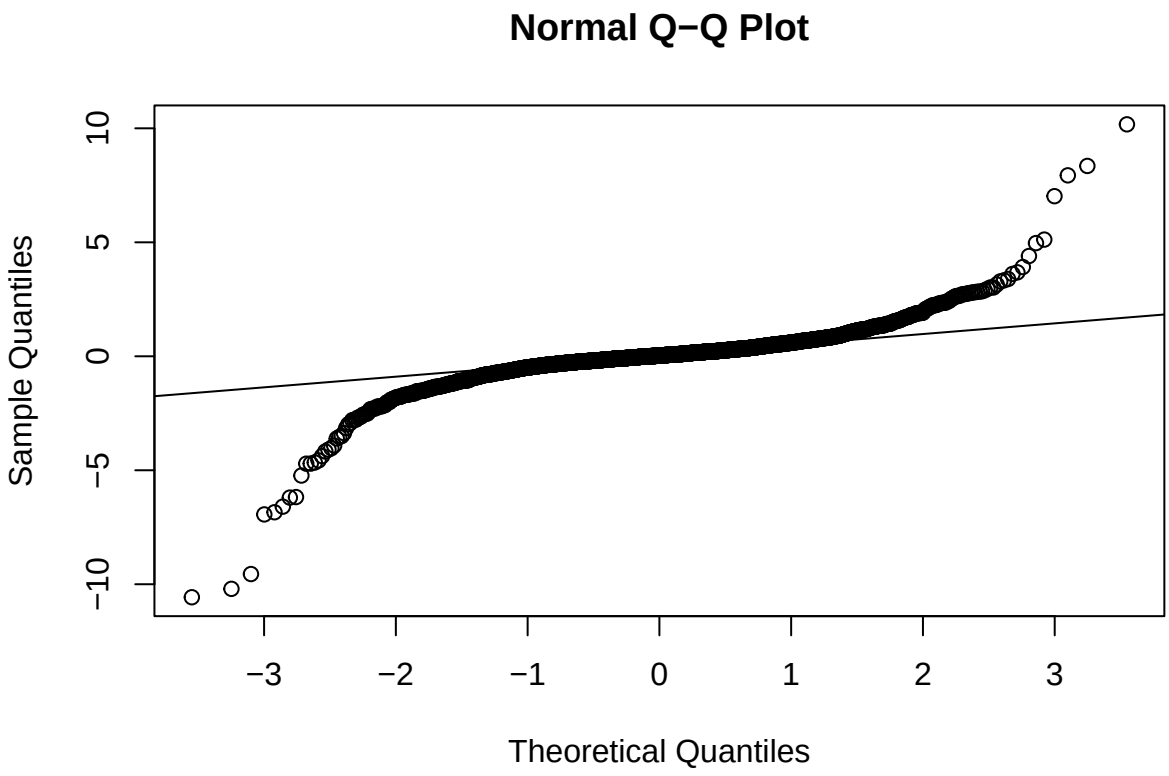

```
tsdiag(ARIMA312)
```







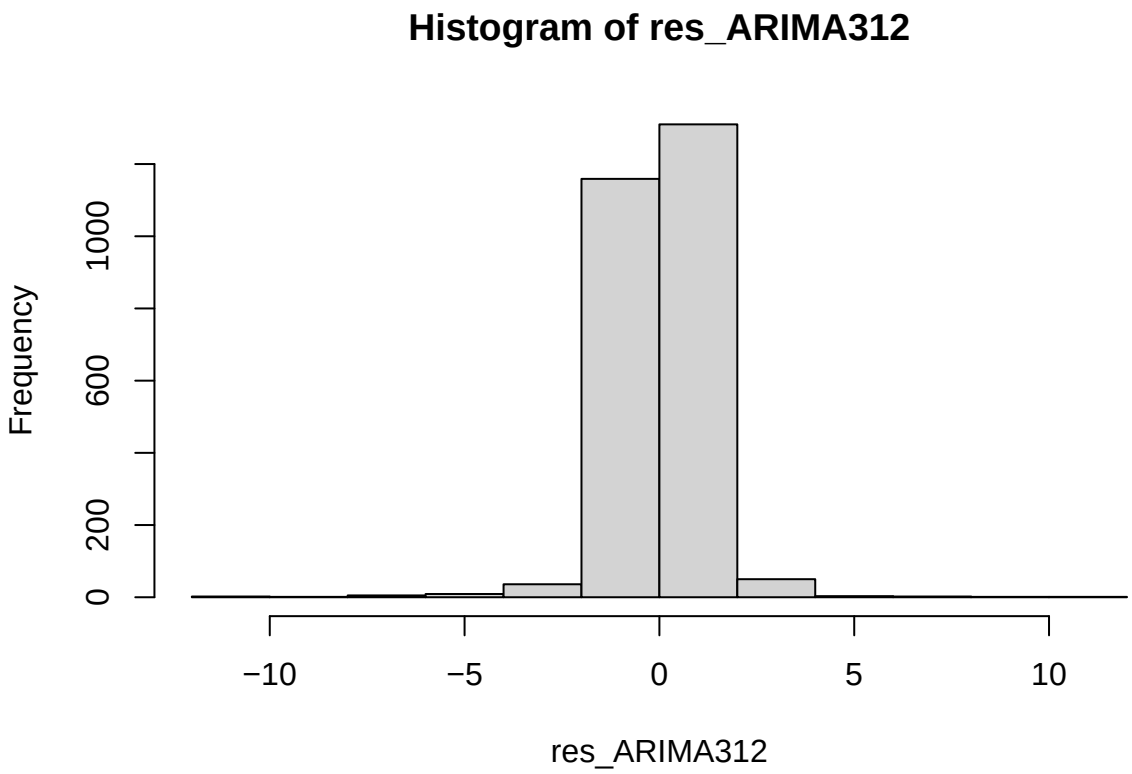

```r
qqnorm(res_ARIMA312)
qqline(res_ARIMA312)
```




```
shapiro.test(res_ARIMA312)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_ARIMA312
## W = 0.76814, p-value < 2.2e-16
```

```
hist(res_ARIMA312)
```

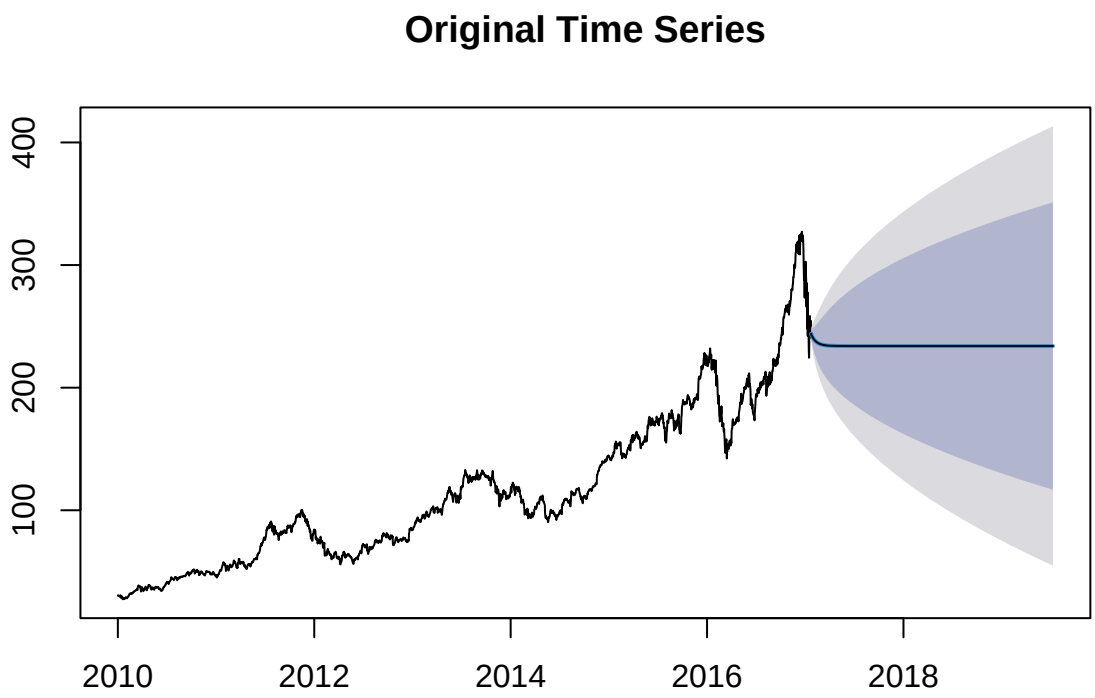


he histogram of the residuals from the ARIMA(3,1,2) model shows that the residuals are relatively normally distributed around zero, with a slight skew to the right. This suggests that the model has done a reasonable job of capturing the patterns in the data, although the presence of skewness might indicate some remaining systematic variation that the model has not fully accounted for.

```
LB.test(ARIMA312, lag=6)
```

```
##
##  Box-Ljung test
##
## data:  residuals from  ARIMA312
## X-squared = 10.698, df = 1, p-value = 0.001073
```

```
forecast_ARIMA312 = forecast(ARIMA312, h=900)
prediction_ARIMA312 = predict(forecast_ARIMA312, n.ahead=20000)
plot(forecast_ARIMA312, main="Original Time Series")
lines(prediction_ARIMA312$mean)
```



The forecast suggests a stable outlook with some uncertainty as indicated by the widening confidence intervals further into the future.

```
# forecast_ARIMA312
```

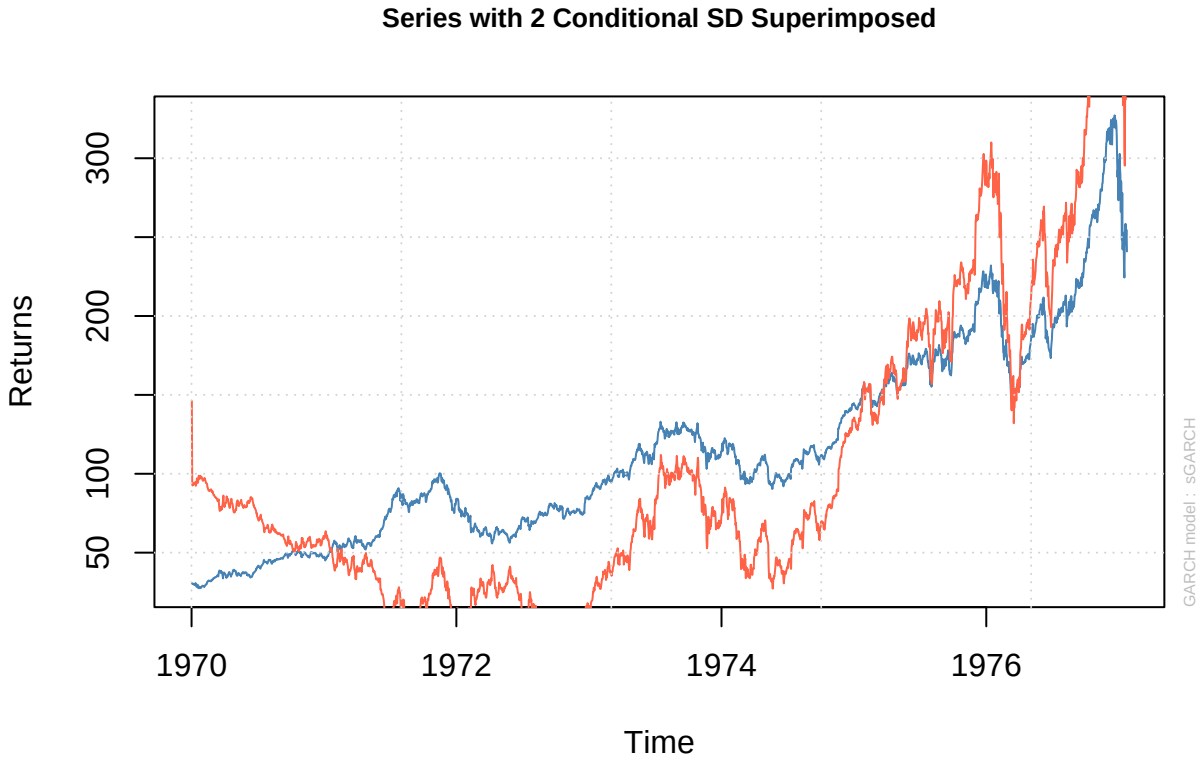
Now, lets try GARCH Model.

```
# Specify the GARCH(1,1) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
                         distribution.model = "norm")

# Fit the GARCH model to the closing prices
fit_garch <- ugarchfit(spec = spec_garch, data = apple_ts)
```

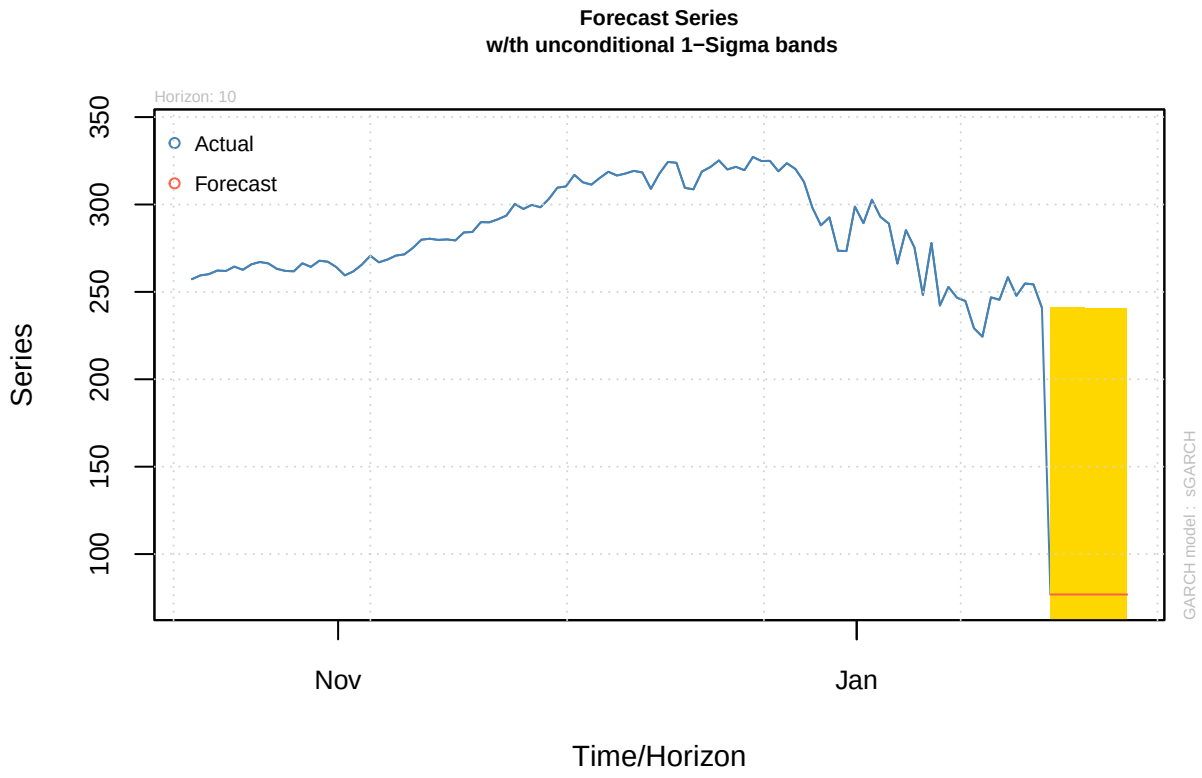
```
summary(fit_garch)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

```
plot(fit_garch, which = 1)
```

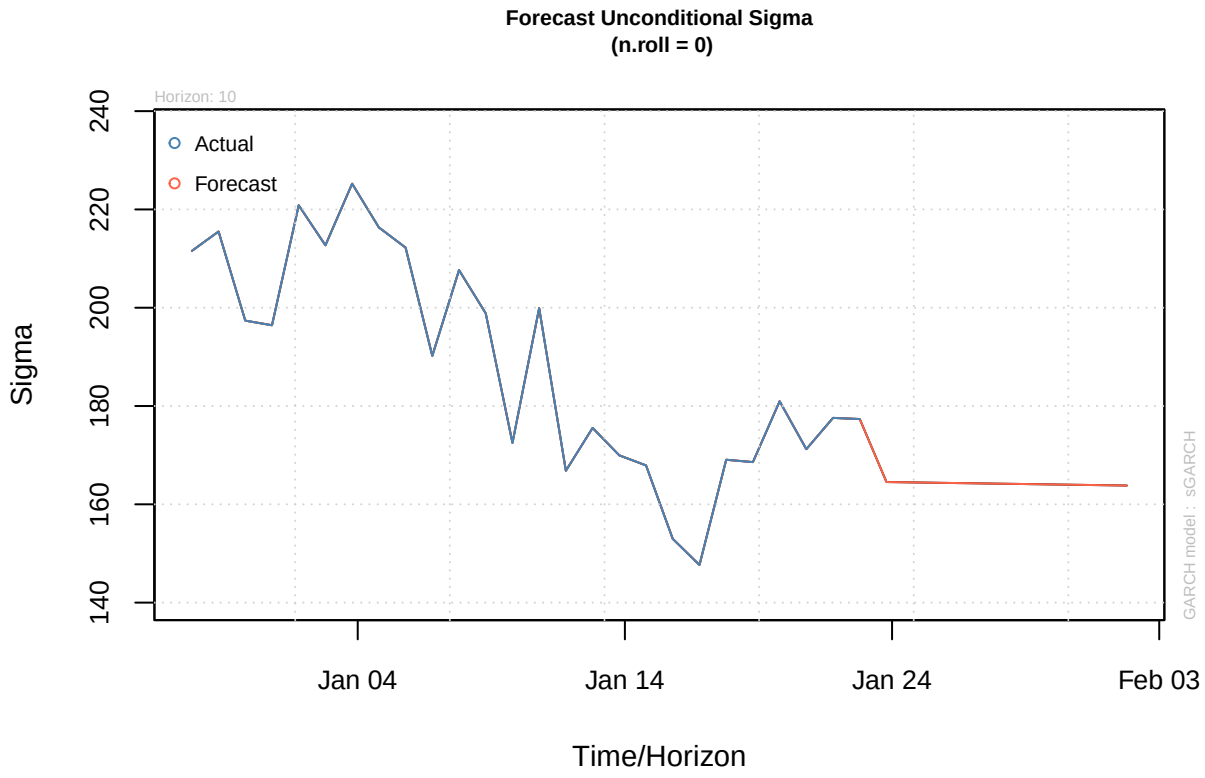



```
forecast_garch <- ugarchforecast(fit_garch, n.ahead = 10)
plot(forecast_garch, which = 1)
```



This shows the actual time series data up until a certain point and then includes a forecast for the next 10 time steps. The forecast region is highlighted in yellow, indicating the forecast values along with their 1-sigma uncertainty bands, providing a visual representation of expected future values and their variability.

```
plot(forecast_garch, which = 3)
```

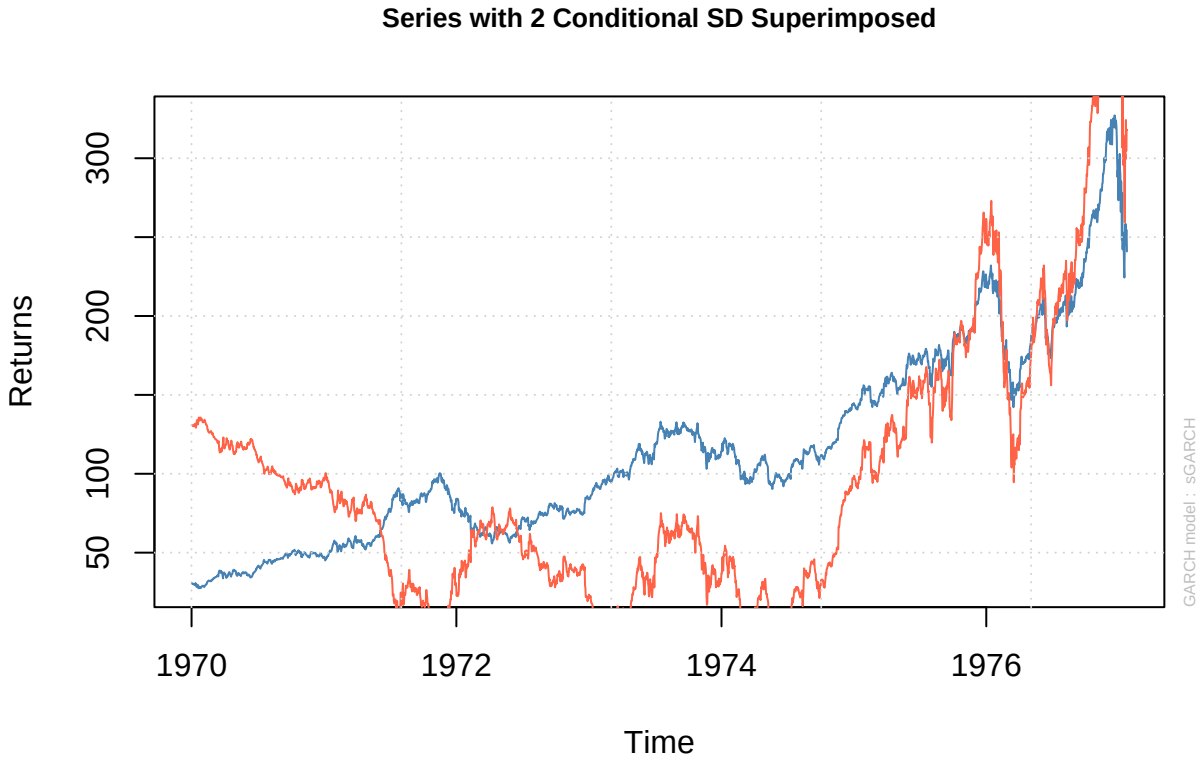



```
# Specify the GARCH(1,3) model
spec_garch <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 3)),
                         mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
                         distribution.model = "norm")

# Fit the GARCH model to the closing prices
fit_garch <- ugarchfit(spec = spec_garch, data = apple_ts)
```

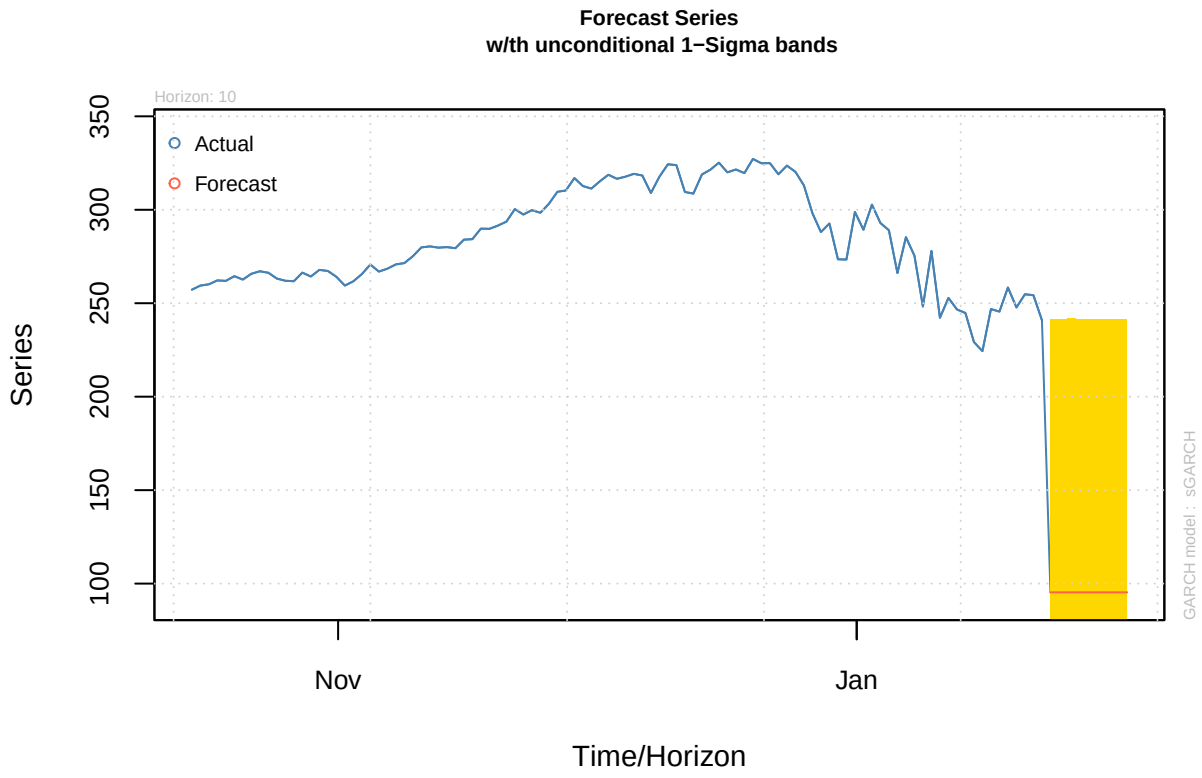
```
summary(fit_garch)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

```
plot(fit_garch, which = 1)
```

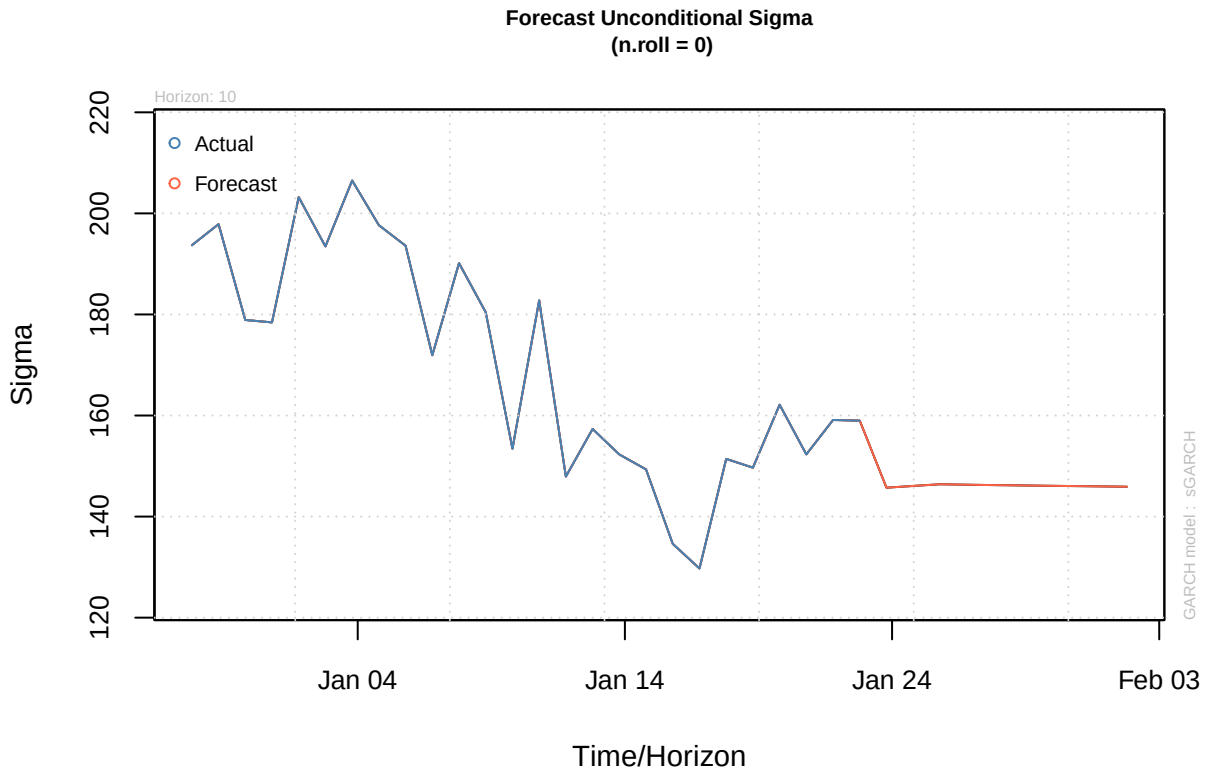



```
forecast_garch <- ugarchforecast(fit_garch, n.ahead = 10)
plot(forecast_garch, which = 1)
```



The graph shows a 10-day forecast using a GARCH(1,3) model, displaying both the actual series and predicted values starting in January. The forecast is marked by red circles and is surrounded by a yellow shaded area representing the 1-Sigma confidence intervals, indicating the probable range of future values based on historical data. This visualizes predicted short-term volatility and is crucial for risk management in financial decision-making.

```
plot(forecast_garch, which = 3)
```



CONCLUSION

The analysis of Apple's stock price using ARIMA and GARCH models highlights different aspects of forecasting stock market behavior. The ARIMA model, typically used for understanding and forecasting data without considering the volatility clustering in stock prices, provides a base prediction over the time series. On the other hand, the GARCH(1,1) and GARCH(1,3) models account for time-varying volatility, capturing the characteristic fluctuations seen in financial markets.

The ARIMA model's forecast shows a continuing trend based on past patterns without giving insight into possible volatility. In contrast, the GARCH models, especially the GARCH(1,3), offer a more nuanced prediction that includes potential volatility, as indicated by the widening confidence intervals in the forecast period. This suggests increased uncertainty in future price movements, reflecting the dynamic and often unpredictable nature of stock prices.

Overall, the combined use of these models provides a comprehensive forecast where ARIMA helps in understanding the direction of the trend while GARCH models add depth by accounting for the volatility dynamics, crucial for making informed investment decisions in the volatile environment of stock markets.