2023 MA641
Time Series Analysis
Semester Project

Time Series Analysis on Seasonal and Nonseasonal datasets

By
Nayanika Ranjan

Department of Mathematical Sciences,
Stevens Institute of Technology,
Hoboken, NJ

Supervisor: Dr. Hadi Safari Katesari

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Abstract

This exploration delves into the nuanced domain of seasonal and non-seasonal time series data, leveraging the capabilities of the R programming language. In the initial segment, the analysis revolves around unraveling monthly sales patterns for Scallops, spanning an extensive 26-year period. The methodologies employed encompass a spectrum of time series models, including Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA), Autoregressive Moving Average (ARMA), Moving Average (MA), and Autoregression (AR). Shifting the focus to another dataset, the analysis navigates the financial landscape by scrutinizing the S&P 500's time series. Extracted from Yahoo Finance, this dataset captures Close prices at three-week intervals. The project's essence lies in providing a comprehensive guide to ARIMA, Autoregressive Conditional Heteroskedasticity (ARCH), and Generalized Autoregressive Conditional Heteroskedasticity (GARCH). The amalgamation of these models seeks to offer a nuanced understanding of their outputs and effectiveness in the intricate domain of time series modeling and forecasting.

[1] Analyzing and Forecasting Scallop Sales

Introduction

This section delves into a comprehensive analysis and forecast of Scallop sales, unraveling intricate patterns, and predicting future trends. The dataset, extracted from the NOAA Fisheries Economics, spans an impressive 26-year duration, meticulously documenting monthly sales figures for Scallops. The primary goal is to leverage advanced time series models, such as Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA), Autoregressive Moving Average (ARMA), Moving Average (MA), and Autoregression (AR). Executed using the R programming language, this initiative aims to provide actionable insights into the complex dynamics of Scallop sales, furnishing a robust foundation for forecasting and informed decision-making.

Dataset Source: https://www.fisheries.noaa.gov/data-tools/fisheries-economics-united-states-data-and-visualizations

Methodology

a) Checking Stationarity:

Before fitting a model, it is crucial to ensure that the time series is stationary. The Dickey-Fuller Test is employed to determine the stationarity of the time series. If the time series is not stationary, methods such as differencing, detrending, or transformation are applied to make it stationary.

b) Finding Models:

Auto-correlation and partial auto-correlation plots are examined to identify potential models.
The orders (p, q) are determined by analyzing the Auto-correlation Function (ACF) and Partial Auto-correlation Function (PACF) plots. The number of lags rising significantly above the confidence interval guides the choice of model parameters. Extended Auto-correlation Function (EACF) can also be utilized to decide on suitable models.

c) Parameter Redundancy & Parameter Estimation:

Once candidate models are identified, the best model is selected based on criteria such as the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and log-likelihood.
Preference is given to models with fewer parameters and superior performance.

d) Residual Analysis:

Residuals play a crucial role in validating the chosen model. ACF plots are used to assess the independence of residuals. Normality of residuals is examined through plots like QQ plots, histograms, and the Shapiro-Wilk Test. The Ljung-Box test determines if the residuals exhibit characteristics of white noise.

e) Forecasting:

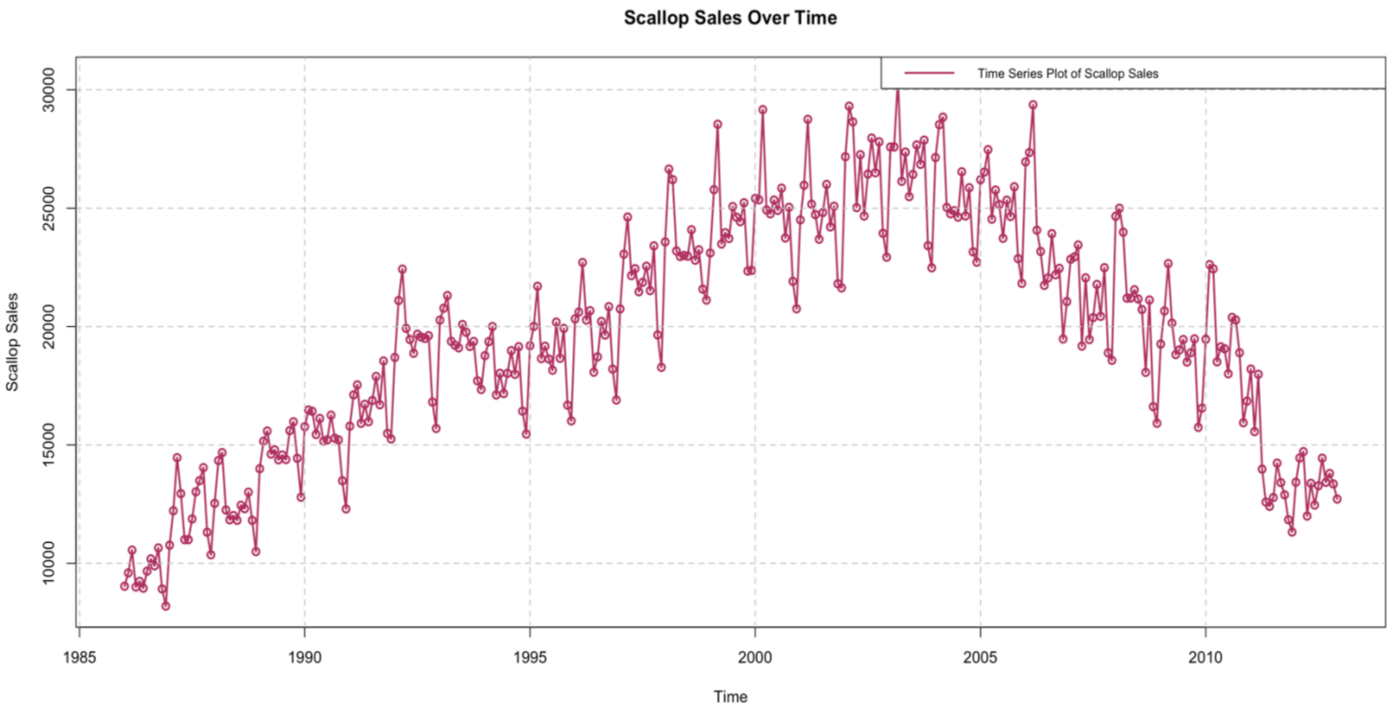
Forecasting represents the final step in time series analysis.
The best-selected model is employed to forecast future values of the original time series, providing insights into how the variable evolves over time or in the future.

Dataset Summary

summary(scallop_ts)

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   8196   15764   19660   19644   23604   30485 
```

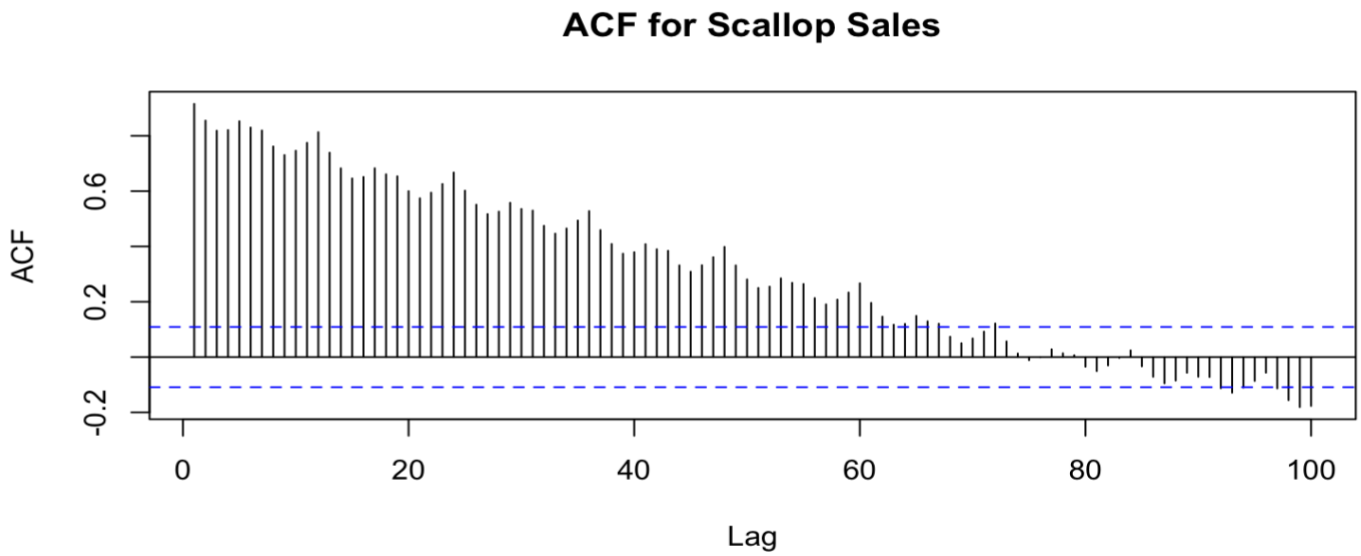
Original time series plot for Scallop sales over the years.

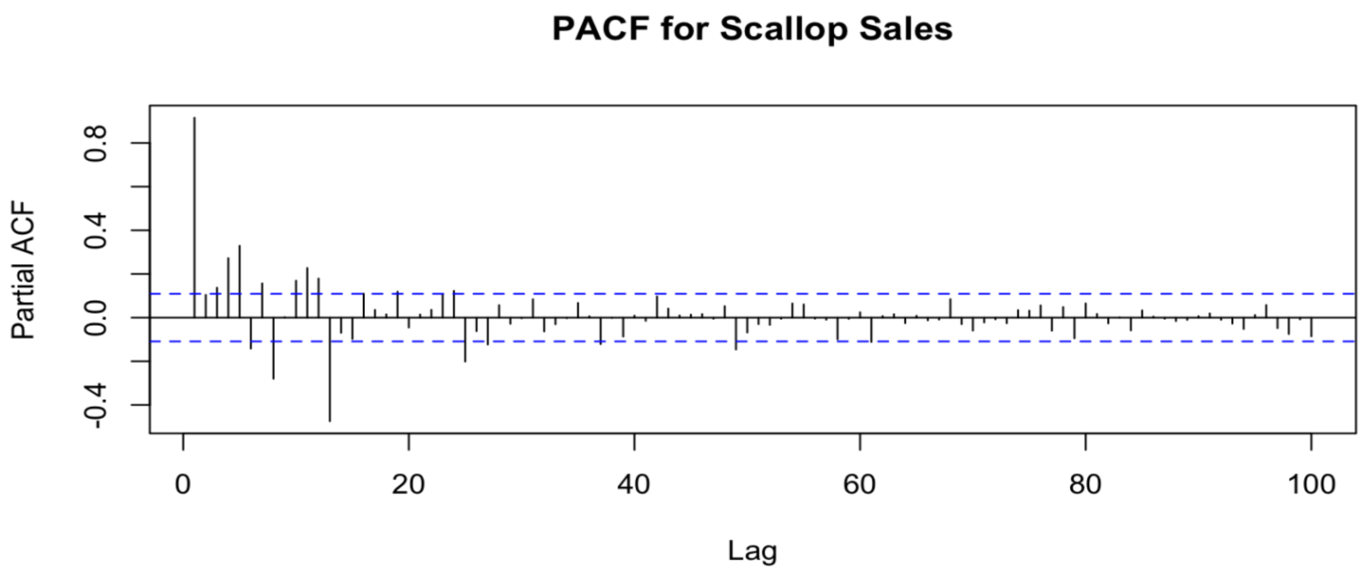



```
        Augmented Dickey-Fuller Test

data:  scallop_ts
Dickey-Fuller = -0.34663, Lag order = 6, p-value = 0.9885
alternative hypothesis: stationary
```

The above time series is not stationary; hence the next steps would be to check the CAF and PACF plots for this and make this series stationary for further analysis.

ACF and PACF plots for Scallops Sales

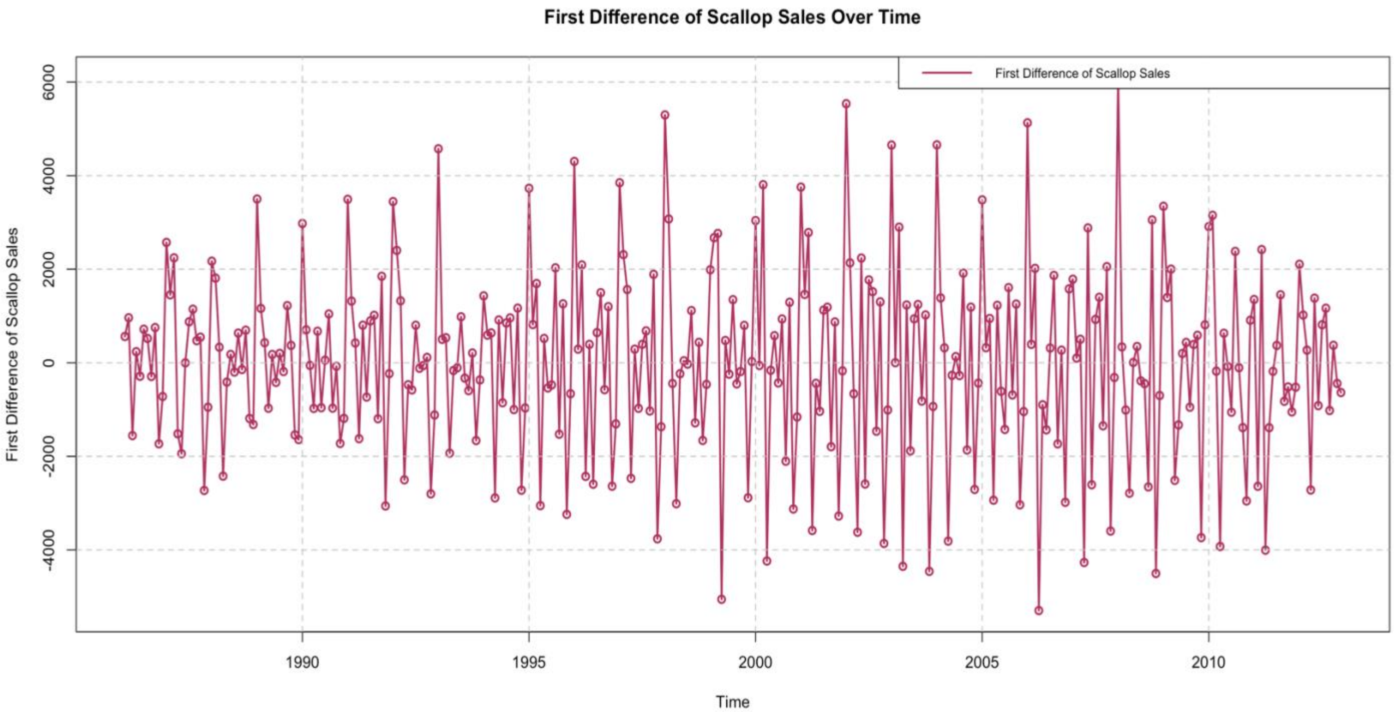




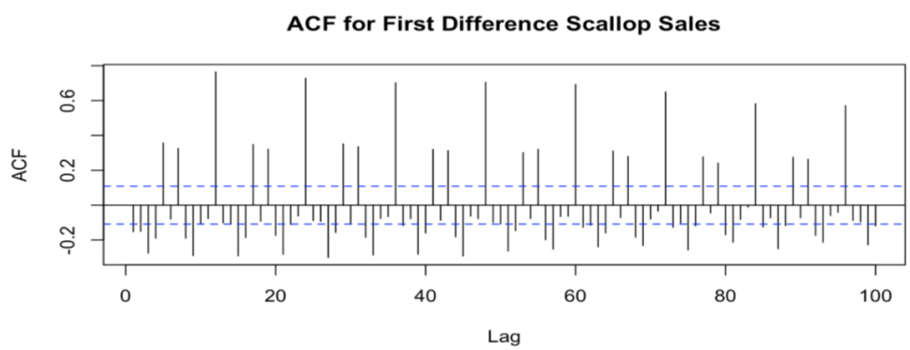
ADF test after differencing the series for stationarity.

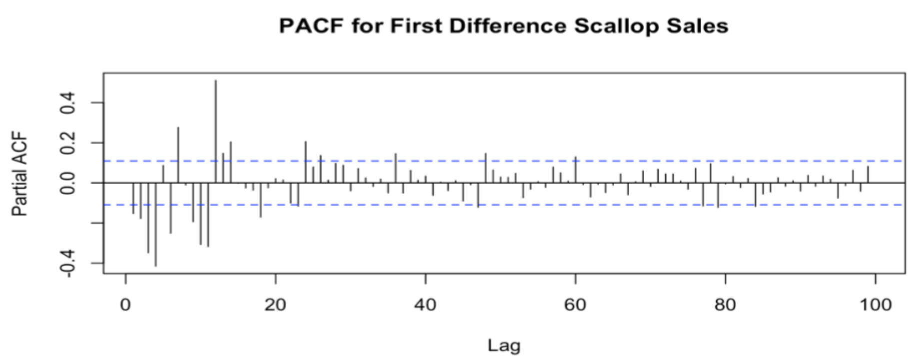
```
     Augmented Dickey-Fuller Test

data:  scallop_diff
Dickey-Fuller = -8.1578, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```

First differenced time series representation of Scallop Sales:



ACF and PACF plots for Scallops Sales



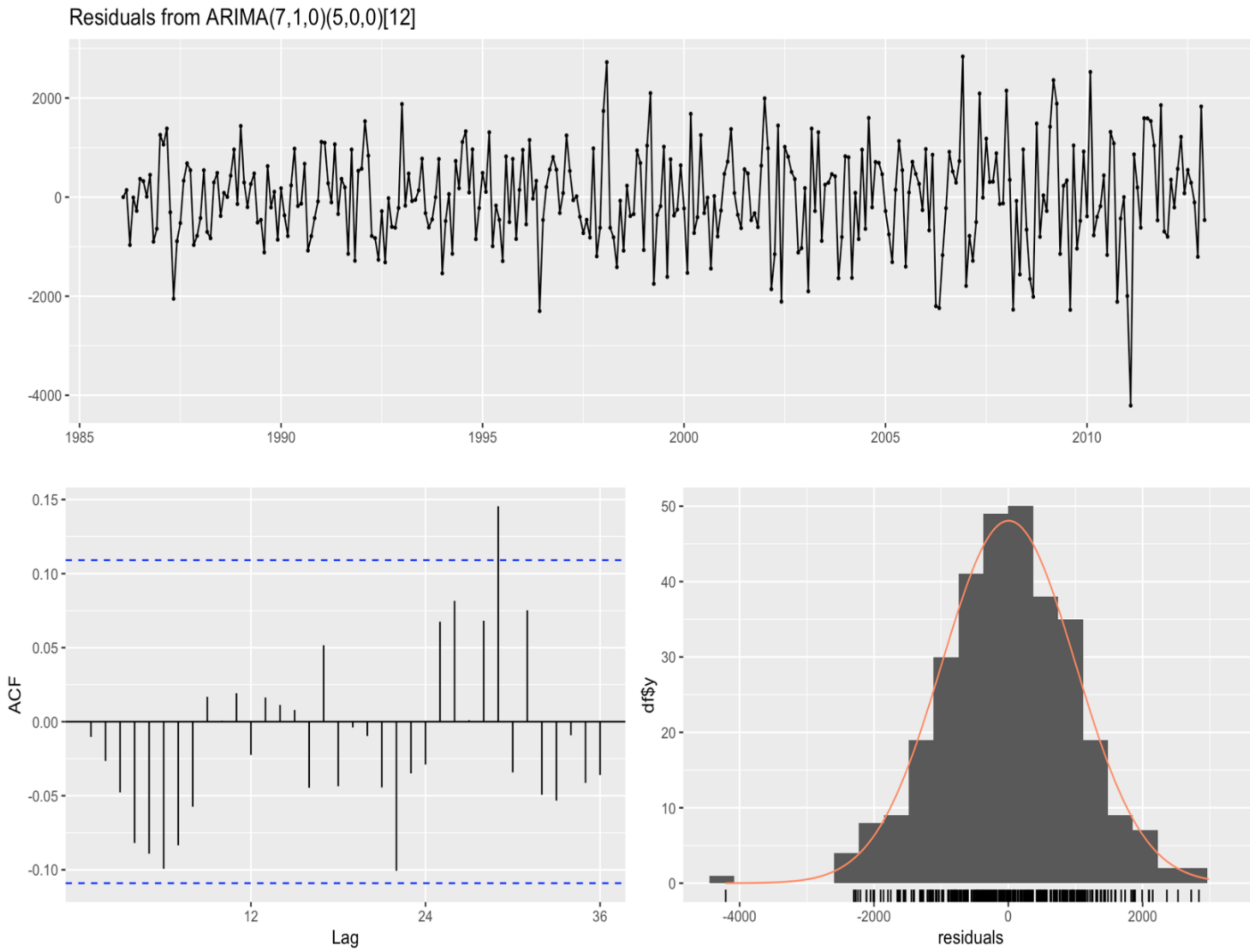


Obtaining the best model based on AIC and BIC values.

```
Best Model (AIC):
$order
[1] 7 1 0
$seasonal_order
[1] 5 0 0
$AIC
[1] 5409.936


Best Model (BIC):
$order
[1] 6 1 0
$seasonal_order
[1] 5 0 0
$BIC
[1] 5456.98
```

```
# # Assuming best_model_aic is the best SARIMA model based on AIC
best_model_aic <- Arima(scallop_diff, order = c(7,1,0), seasonal = list(order = c(5,0,0), period = 12))
# par(mfrow = c(1, 1))
# plot(best_model_aic)

set.seed(123)
par(mfrow = c(1, 1))
# Residual diagnostics
checkresiduals(best_model_aic)
```

Residual analysis of the selected model is shown below. The histogram shows a normal curve and QQ plot seems like a good fit as well.



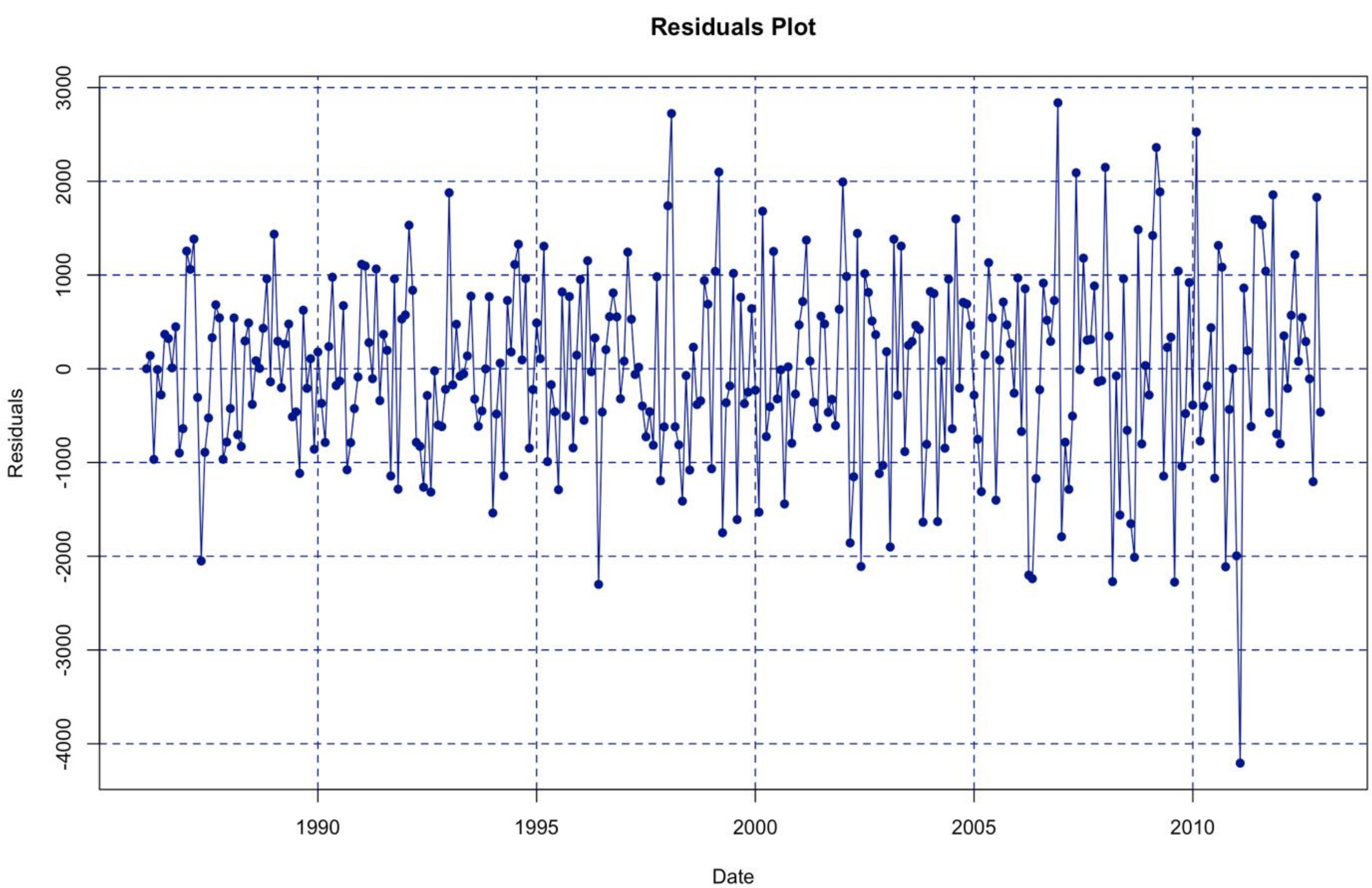

```
	Ljung-Box test

data:  Residuals from ARIMA(7,1,0)(5,0,0)[12]
Q* = 20.301, df = 12, p-value = 0.0616

Model df: 12.   Total lags used: 24
```

```
# Create a QQ plot for SARIMA(7,1,0)x12(5,0,0)
set.seed(123)
par(mfrow = c(1, 1))
residuals <- residuals(best_model_aic)
```

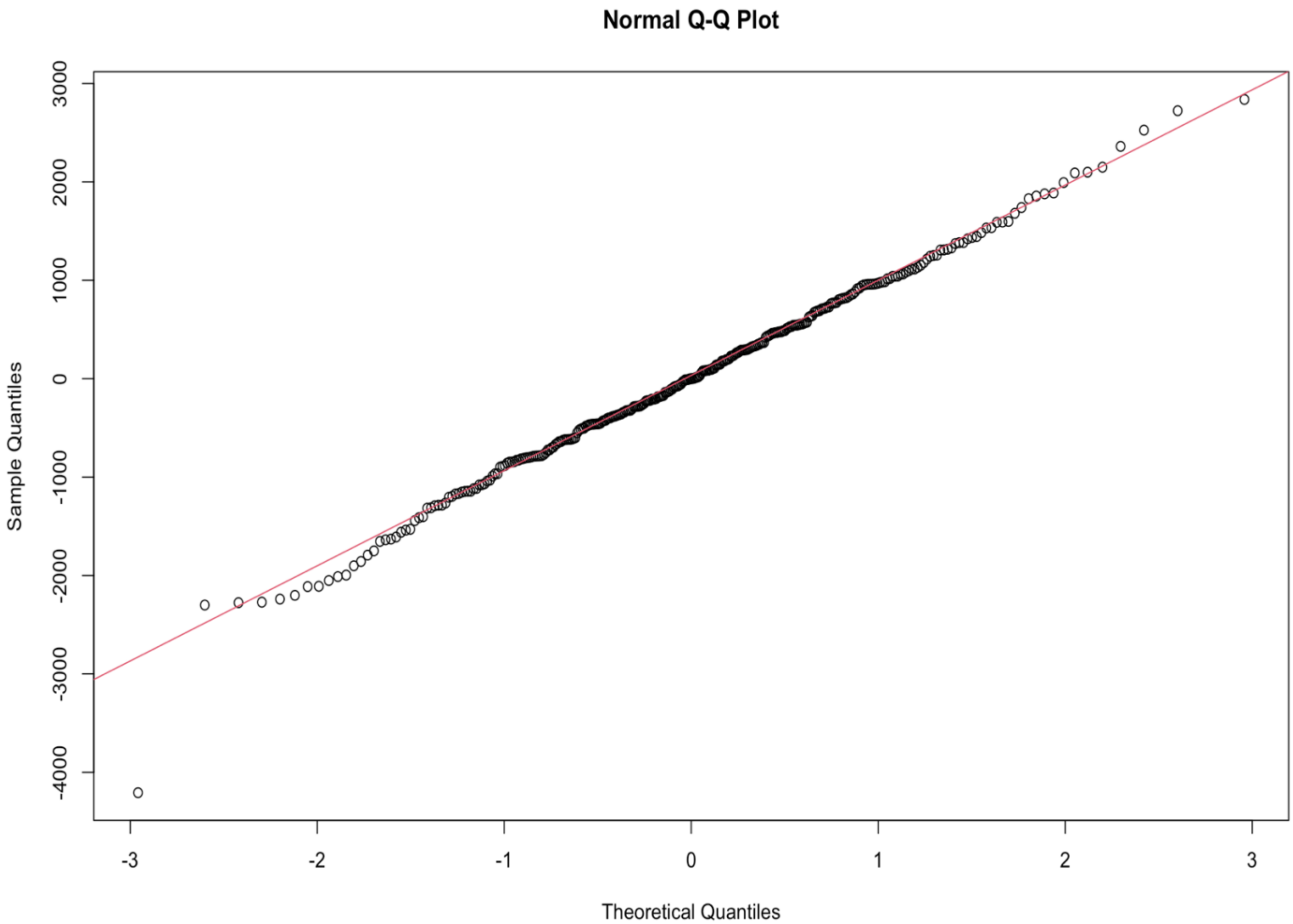
#Shapiro Test to check normality
print(shapiro.test(residuals))

```
     Shapiro-Wilk normality test

data:  residuals
W = 0.99359, p-value = 0.1873
```

# Assuming 'residuals' is your vector of residuals
plot(residuals, type='o', col='darkblue', pch=16, xlab='Date', ylab='Residuals', main='Residuals Plot')
grid(col = 'darkblue', lty = 2)

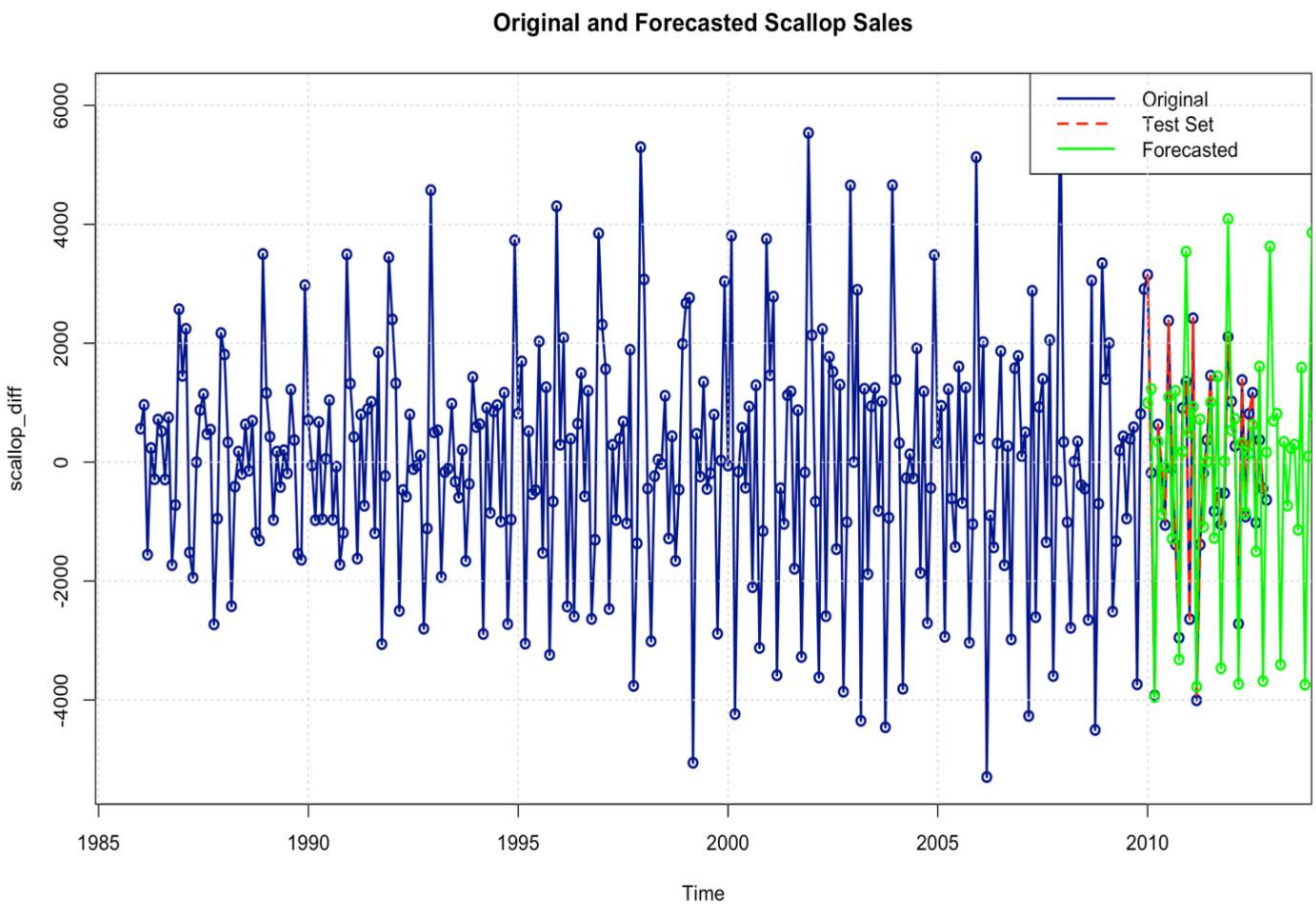


```
qqnorm(residuals)
qqline(residuals, col = 2)
```




```
# Reset the plotting layout
par(mfrow = c(1, 1))
# Ensure the time index is in order
scallop_diff <- ts(scallop_diff, start = c(1986, 1), frequency = 12)

# Define training set and test set
train_set <- window(scallop_diff, end = c(2009, 12))
test_set <- window(scallop_diff, start = c(2010, 1))
```

```
# ADF test to check for stationary
adf_result <- adf.test(test_set)
## Warning in adf.test(test_set): p-value smaller than printed p-value
adf_result

     Augmented Dickey-Fuller Test

data:  test_set
Dickey-Fuller = -5.4108, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary
```



The forecasted Scallop Sales can be seen in the above plot in green colour, along with the original plot in blue and the test values in red.

[2] Analyzing and Forecasting S&P500 Index Movement

Introduction

This segment embarks on an extensive exploration and projection of the S&P500 index movement, unraveling intricate patterns and predicting future trends. The dataset, sourced from Yahoo Finance, encapsulates a rich historical perspective of the S&P500 over a substantial timeframe. Monthly index values have been meticulously recorded, spanning multiple years. The primary objective is to harness advanced time series modeling techniques, including Autoregressive Integrated Moving Average (ARIMA), and in cases where traditional models fall short, alternative approaches such as Generalized Autoregressive Conditional Heteroskedasticity (GARCH) are explored. Executed using the R programming language, this initiative seeks to offer actionable insights into the complex dynamics of the S&P500 index, laying the groundwork for forecasting and strategic decision-making.

Dataset Source: [Yahoo Finance - S&P500 Historical Data]

Methodology

a) Checking Stationarity:

To ensure the robustness of the models, a crucial initial step involves checking the stationarity of the time series. The Augmented Dickey-Fuller (ADF) Test is employed to ascertain stationarity. If non-stationarity is detected, methods such as differencing are applied to transform the data into a stationary format.

b) Basic Exploratory Data Analysis (EDA):

An in-depth examination of the data is conducted to identify trends, patterns, and potential outliers. Descriptive statistics and visualizations are employed to gain insights into the historical behavior of the S&P500 index.

c) Finding Models:

Auto-correlation and partial auto-correlation plots are utilized to identify potential models. Orders (p, q) are determined by analyzing Auto-correlation Function (ACF) and Partial Auto-correlation Function (PACF) plots. The suitability of models is assessed based on the characteristics observed in the extended Auto-correlation Function (EACF).

d) Model Selection and Evaluation:

Model selection involves employing criteria such as the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and log-likelihood to choose the best-performing model. In instances where traditional models like ARIMA fall short, alternative models such as GARCH are explored.

e) Residual Analysis:

Residuals are scrutinized to validate the chosen model. Diagnostic tools such as ACF plots, QQ plots, histograms, and statistical tests like the Ljung-Box test are employed to assess the adequacy of the chosen model.

f) Forecasting:

Forecasting is the culmination of the analysis, where the selected model is utilized to project future values of the S&P500 index. This step provides valuable insights into how the index may evolve over time, facilitating informed decision-making in financial planning and investment strategies.

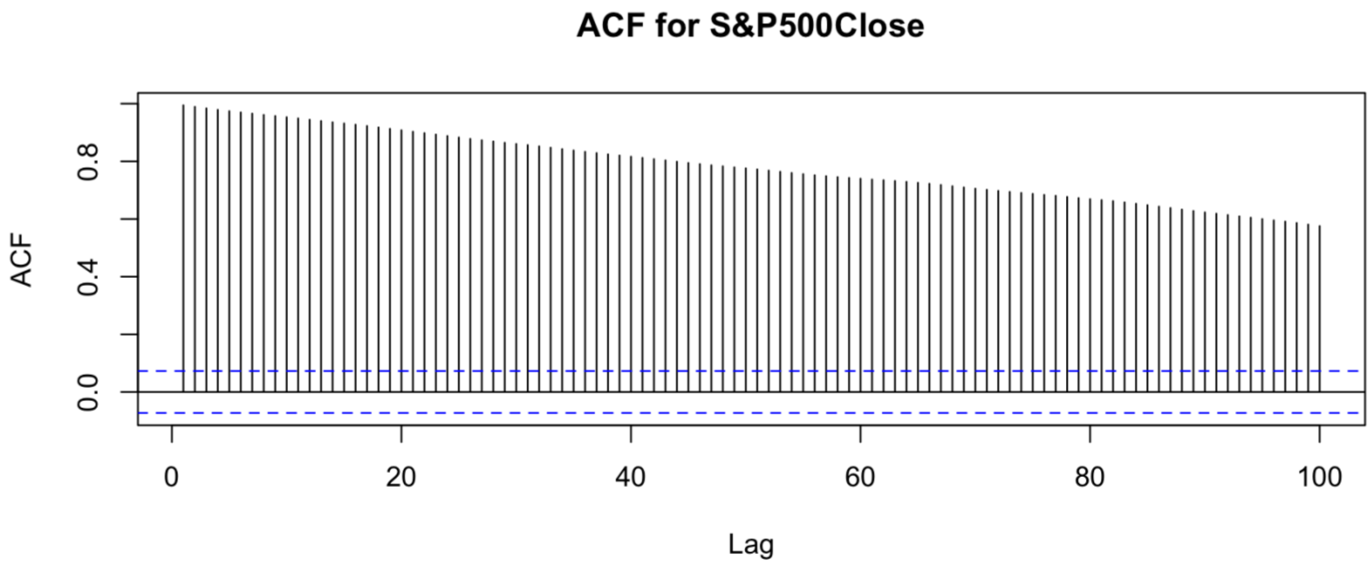
# Check the structure and summary of the data
summary(sp500)

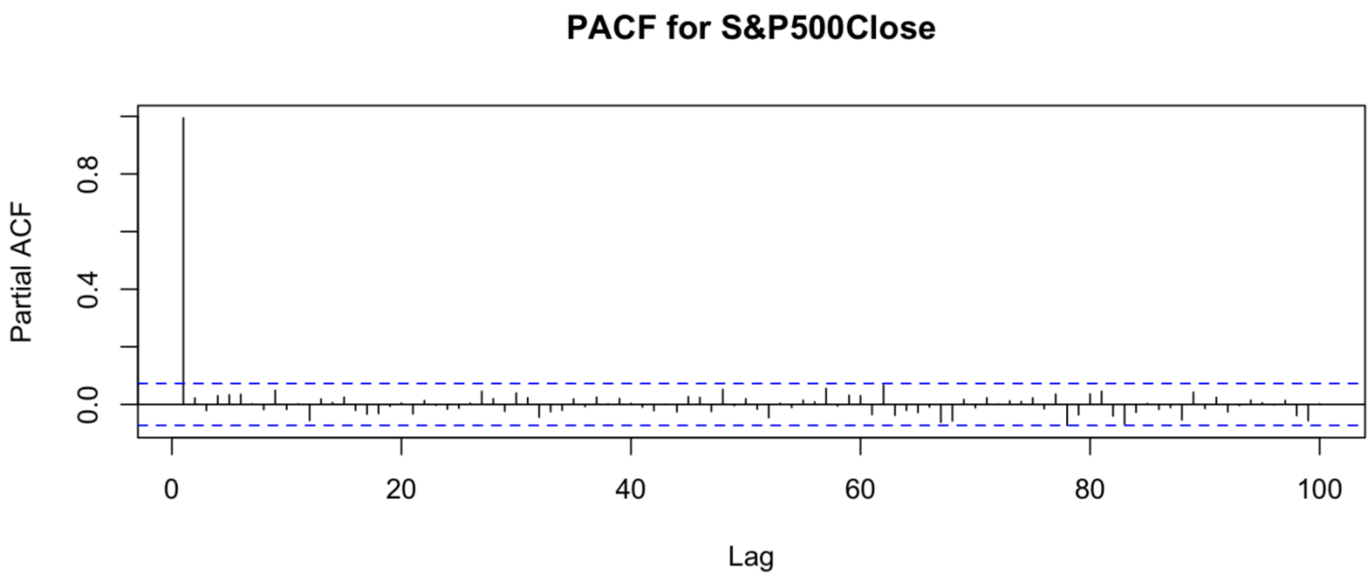
```
      Date                 Open           High           Low            Close     
 Min.   :2010-01-01   Min.   :1028   Min.   :1071   Min.   :1011   Min.   :1027  
 1st Qu.:2013-06-22   1st Qu.:1641   1st Qu.:1662   1st Qu.:1625   1st Qu.:1650  
 Median :2016-12-12   Median :2251   Median :2273   Median :2213   Median :2255  
 Mean   :2016-12-12   Mean   :2520   Mean   :2556   Mean   :2482   Mean   :2524  
 3rd Qu.:2020-06-03   3rd Qu.:3234   3rd Qu.:3279   3rd Qu.:3211   3rd Qu.:3246  
 Max.   :2023-11-24   Max.   :4775   Max.   :4819   Max.   :4734   Max.   :4779  

   Adj.Close        Volume         
 Min.   :1027   Min.   :5.038e+09  
 1st Qu.:1650   1st Qu.:1.627e+10  
 Median :2255   Median :1.860e+10  
 Mean   :2524   Mean   :1.900e+10  
 3rd Qu.:3246   3rd Qu.:2.069e+10  
 Max.   :4779   Max.   :4.123e+10  
```

# Checking the "Date" column is in Date format
sp500Date <- as.Date(sp500$Date)
sp500Close <- sp500$Close

The original S&P500 time series was plotted for the Close Price as represented below.

ACF and PACF are shown below for the original time series.
ACF shows a decreasing trend in general but no clear MA model aspects. Whereas there's one significant lag in PACF, and some sort of pattern afterwards.





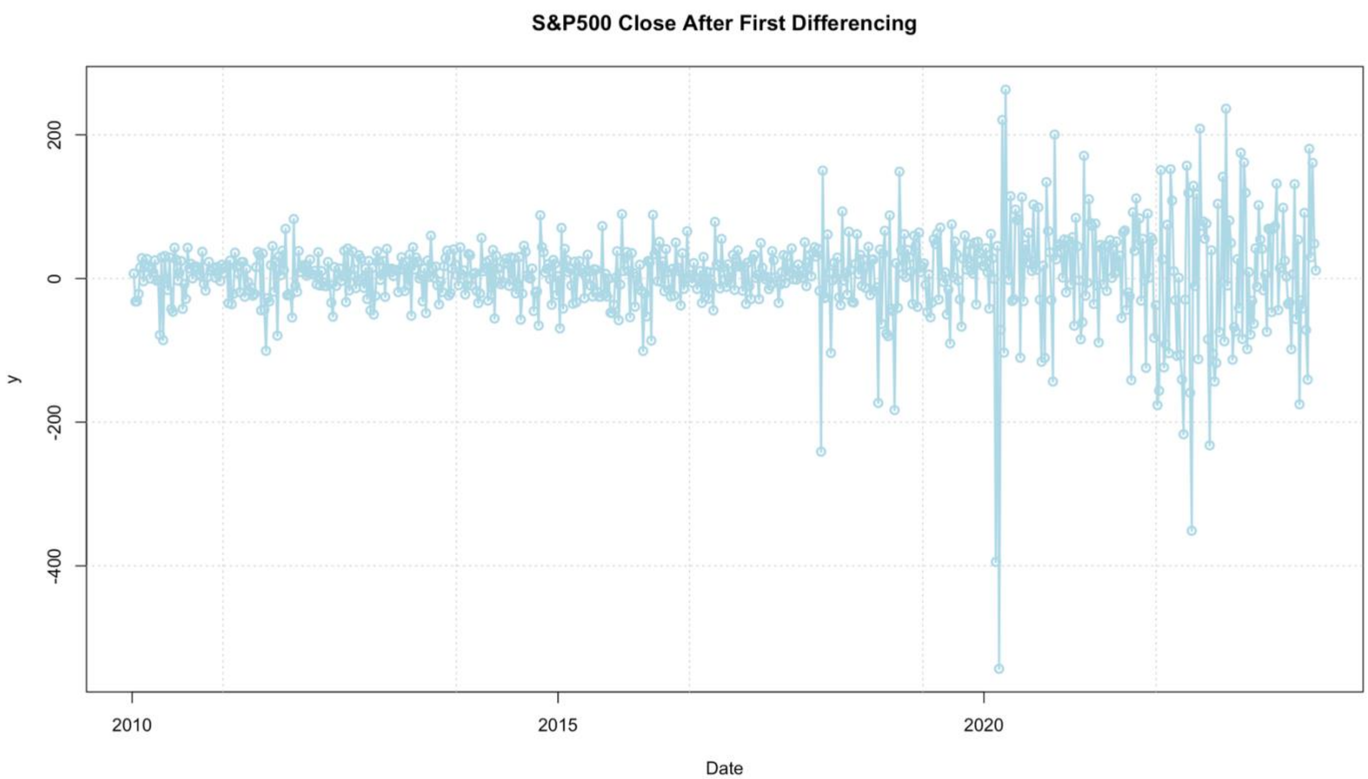
ADF test for the original mode, showed that the series is not stationary.

```
	Augmented Dickey-Fuller Test

data:  sp500Close
Dickey-Fuller = -2.4036, Lag order = 8, p-value = 0.4075
alternative hypothesis: stationary
```

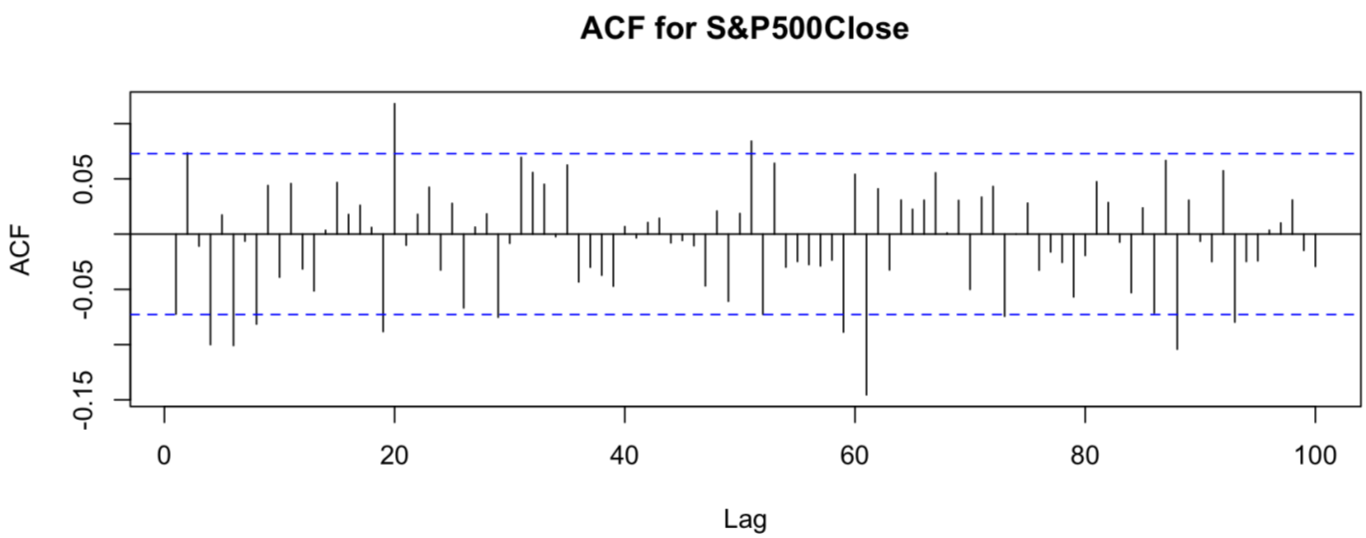
Performing ADF test again on the differenced time series of S&P500.

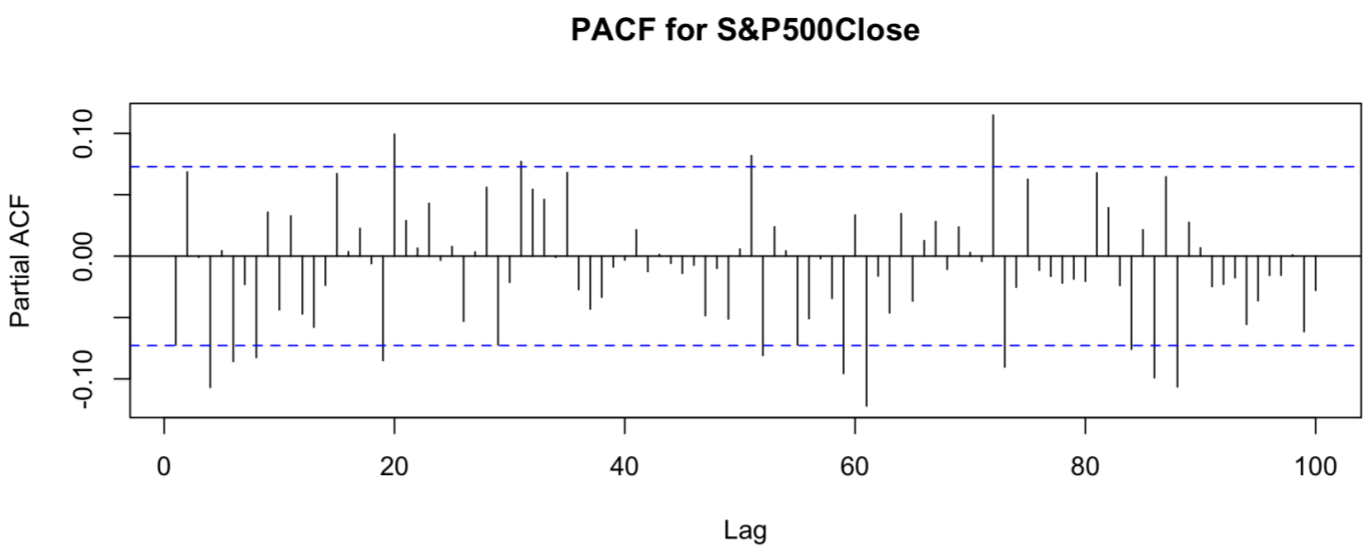
```
     Augmented Dickey-Fuller Test

data:  sp500_close_diff
Dickey-Fuller = -10.16, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary
```

Now that the series is stationary, below if the first differencing representation of it.



ACF and PACF of the differenced time series look almost alike, suggesting correlation between the two.





EACF values for the differenced series

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

Based on the above values, ARIMA model was applied to the differenced time series.

```
Series: sp500Close
ARIMA(2,1,4) with drift

Coefficients:
          ar1     ar2     ma1      ma2     ma3      ma4   drift
      -0.1058  0.5798  0.0425  -0.5324  0.0304  -0.1622  4.6795
s.e.   0.1256  0.1174  0.1270   0.1148  0.0403   0.0366  1.6457

sigma^2 = 3810:  log likelihood = -4014.28
AIC=8044.56   AICc=8044.76   BIC=8081.25
```

# Summary of the ARIMA model
summary(arima_model)


```
Series: sp500Close
ARIMA(2,1,4) with drift

Coefficients:
          ar1     ar2     ma1      ma2     ma3      ma4   drift
      -0.1058  0.5798  0.0425  -0.5324  0.0304  -0.1622  4.6795
s.e.   0.1256  0.1174  0.1270   0.1148  0.0403   0.0366  1.6457

sigma^2 = 3810:  log likelihood = -4014.28
AIC=8044.56   AICc=8044.76   BIC=8081.25
```



```
Training set error measures:
                      ME     RMSE      MAE         MPE     MAPE      MASE
Training set -0.03436625 61.38728 39.47542 -0.07512502 1.549377 0.9758678
                     ACF1
Training set -0.001927647
```

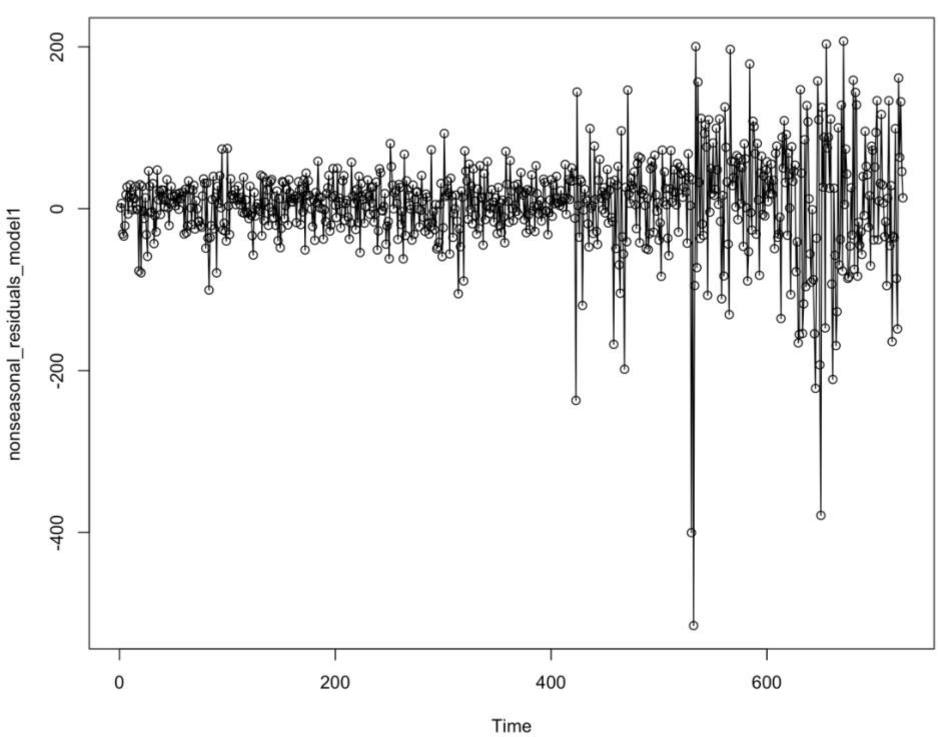
# Print optimization output
print(arima_model$optim.output)
NULL

As the optimization output is NULL, it means that the path followed might a good fit.

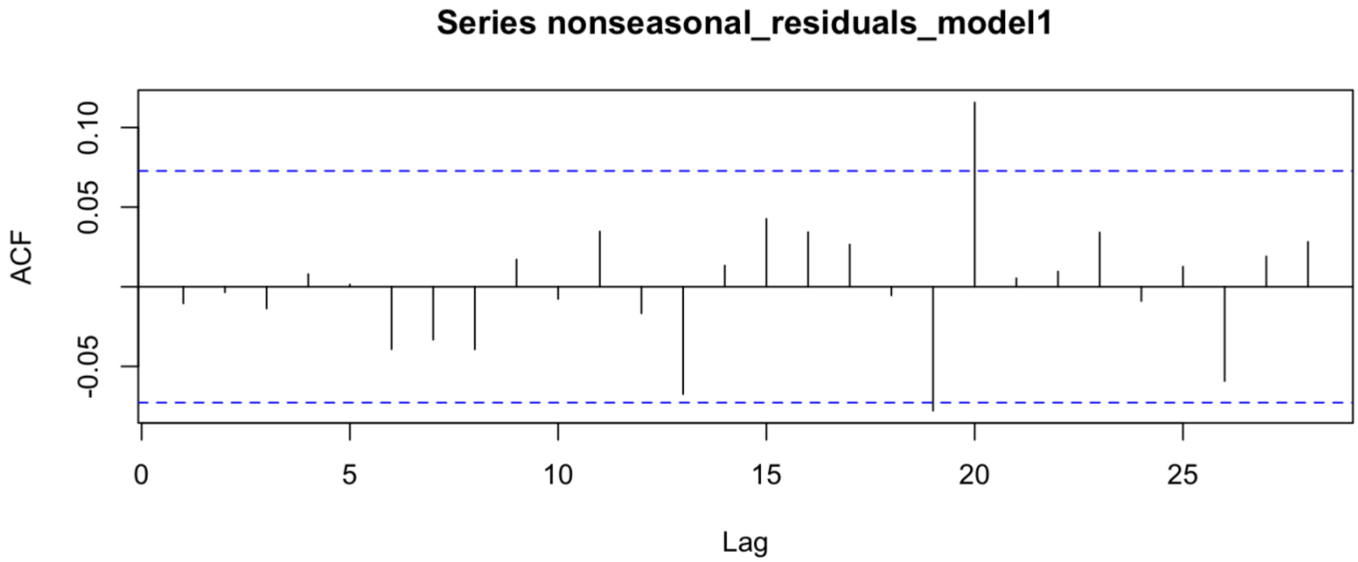
# Manual ARIMA with specified order (2,1,4)
ns_model1 = arima(sp500Close, order=c(2,1,4))
print(ns_model1)

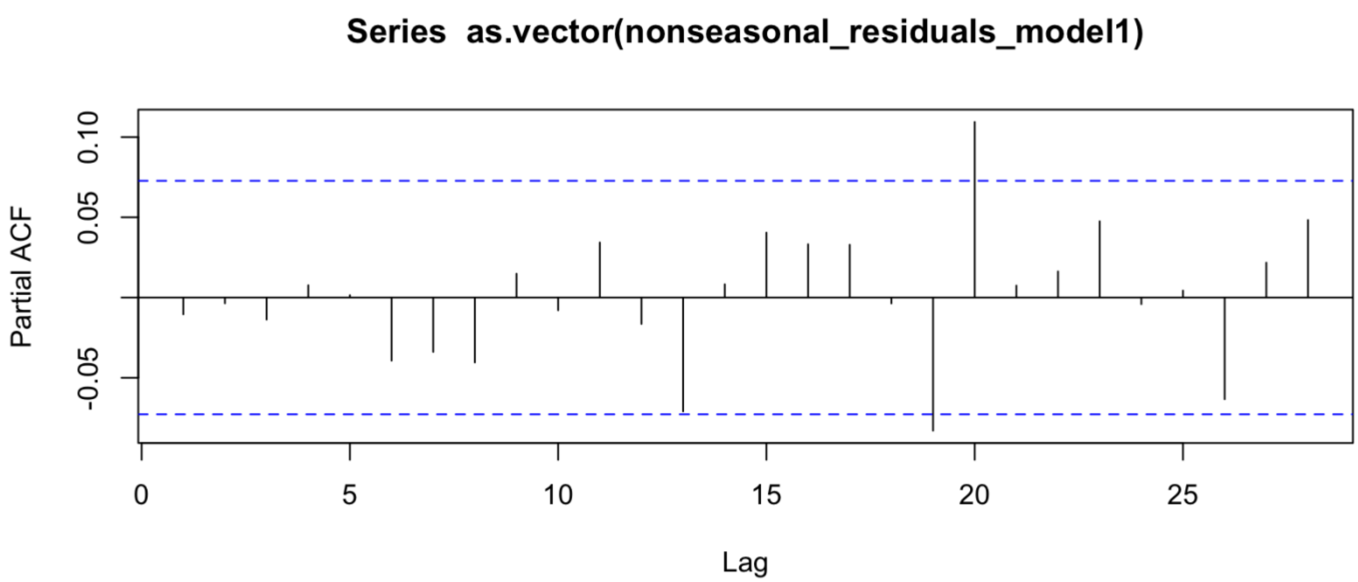
```
Call:
arima(x = sp500Close, order = c(2, 1, 4))

Coefficients:
          ar1     ar2     ma1      ma2     ma3      ma4
      -0.1472  0.5463  0.0935  -0.4907  0.0365  -0.1538
s.e.   0.1514  0.1399  0.1522   0.1354  0.0392   0.0361

sigma^2 estimated as 3811:  log likelihood = -4017.84,  aic = 8047.68
```

Residuals of the ARIMA(2,1,4) is shown below:

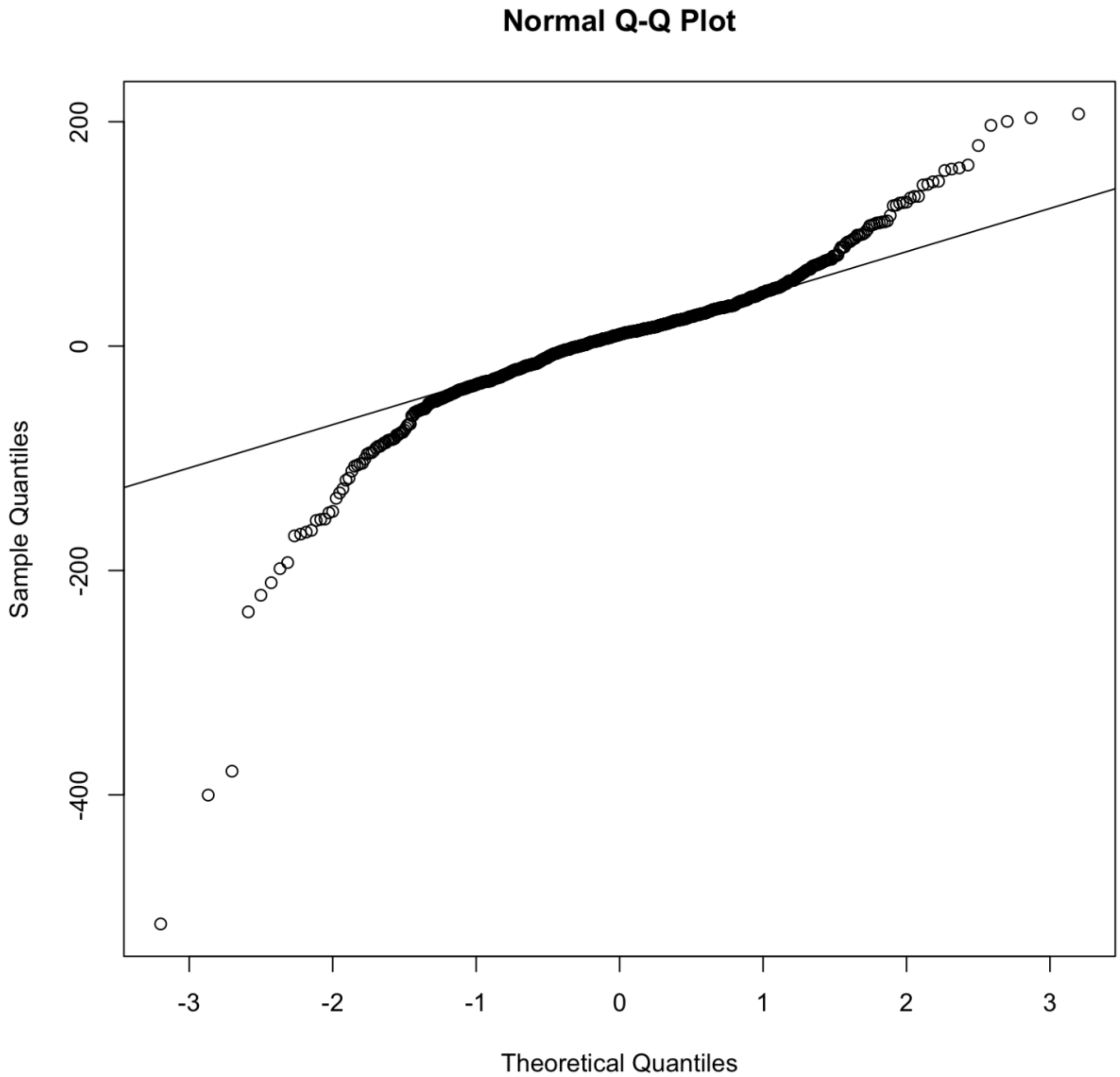


ACF and PACF of residuals

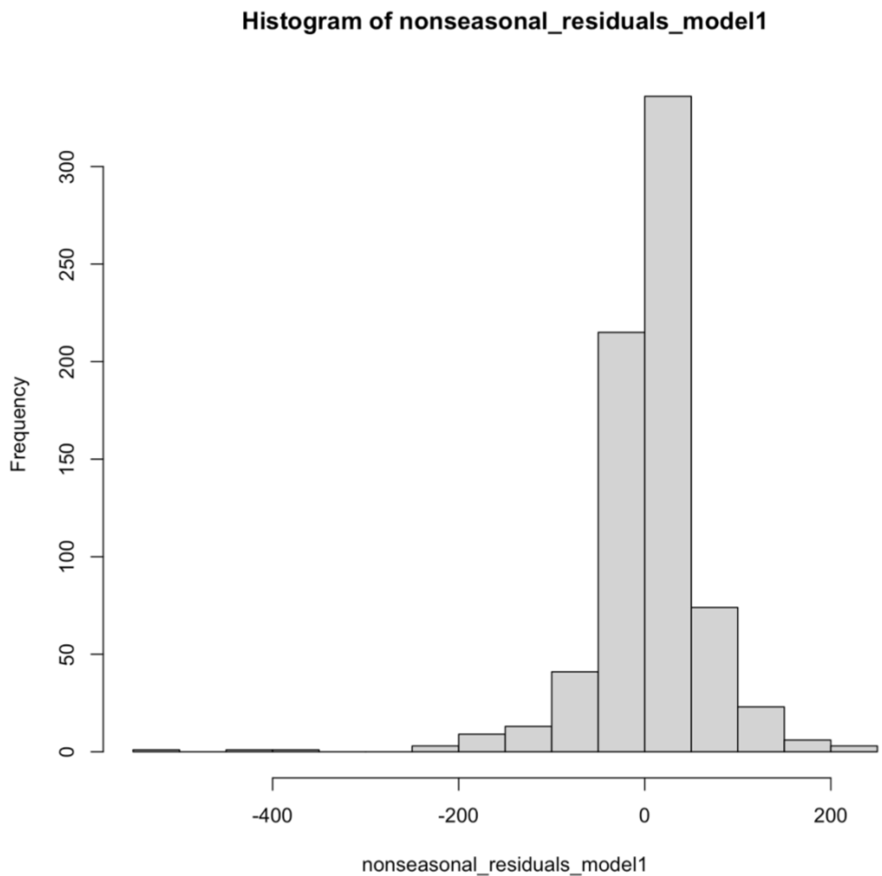




Below QQ plot of residuals shows that most of the data points are covered in the model fitting, but still leaving out significant ones.



# Histogram of residuals
hist(nonseasonal_residuals_model1)

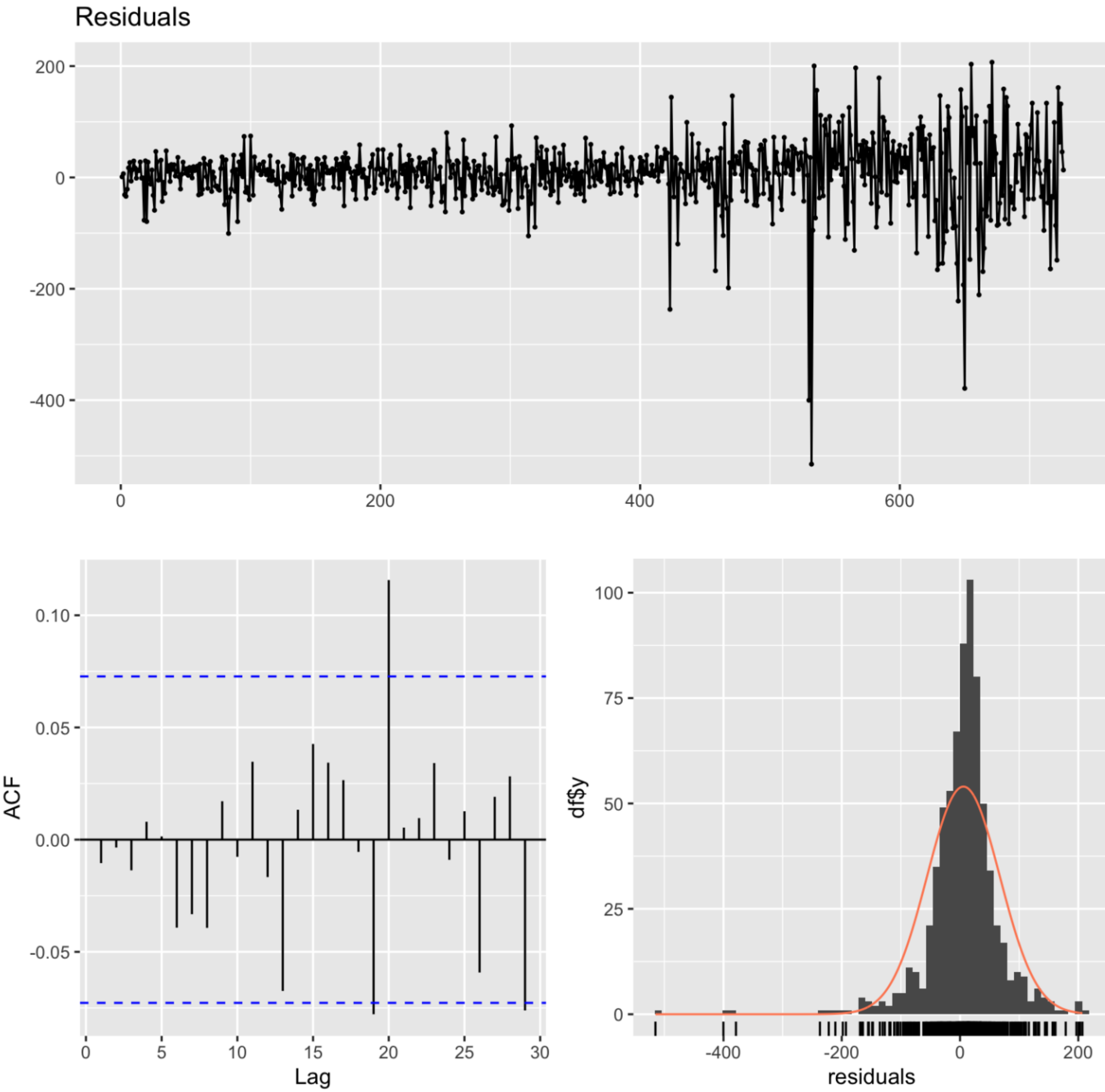


# Shapiro-Wilk normality test
print(shapiro.test(nonseasonal_residuals_model1))

```
     Shapiro-Wilk normality test

data:  nonseasonal_residuals_model1
W = 0.86461, p-value < 2.2e-16
```

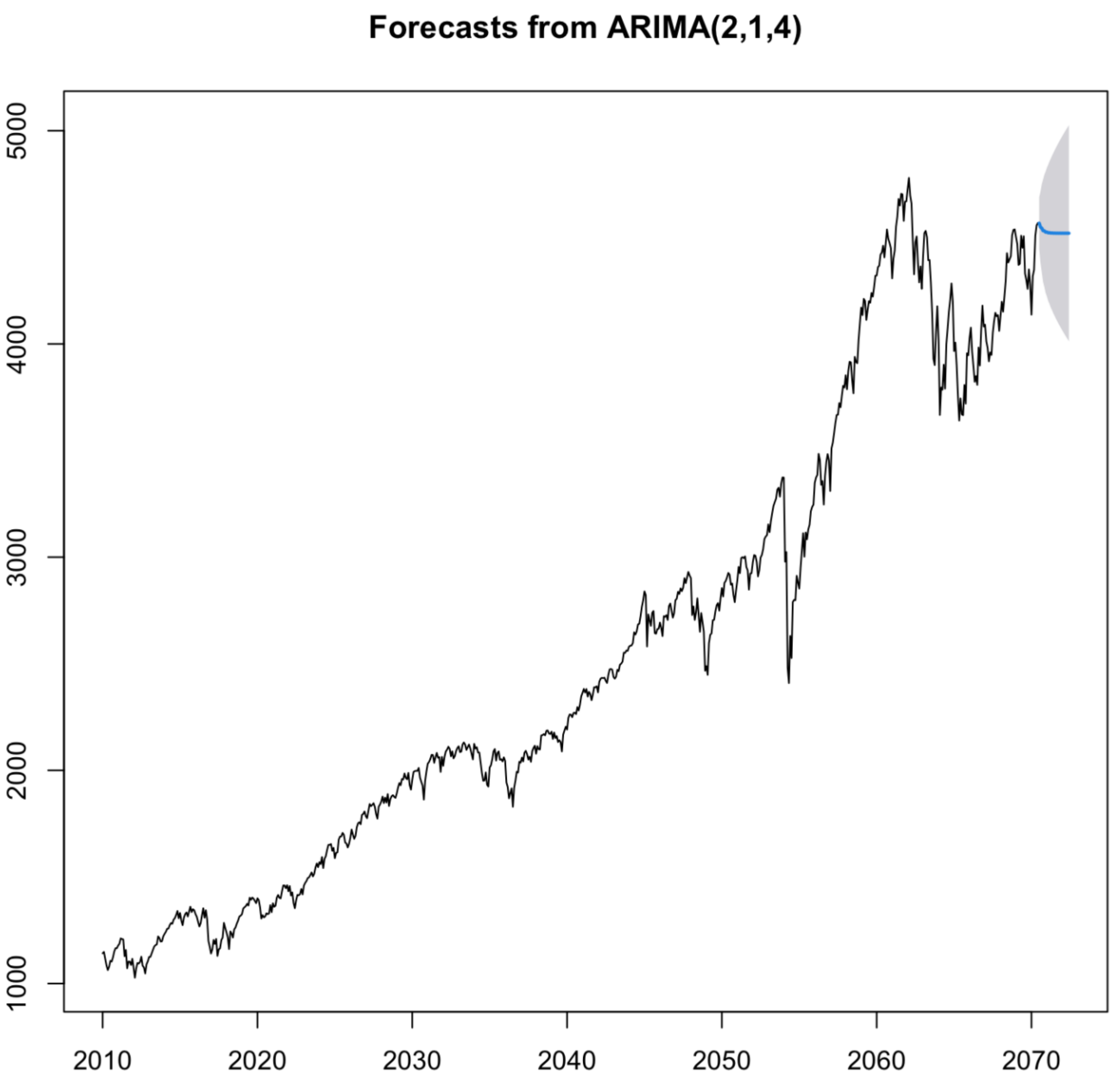
# Check residuals using checkresiduals function
checkresiduals(nonseasonal_residuals_model1)

```
     Ljung-Box test

data:  Residuals
Q* = 3.6087, df = 10, p-value = 0.9633

Model df: 0.   Total lags used: 10
```




```
# Reset plotting layout
par(mfrow = c(1, 1))
```

```
# Forecast using ARIMA model
model1 <- arima(ts(sp500Close, frequency = 12, start = c(2010, 1)), order = c(2, 1, 4))
model1$x <- ts(sp500Close, frequency = 12, start = c(2010))
f = forecast(model1, level = c(95), h = 24)
plot(f)
```

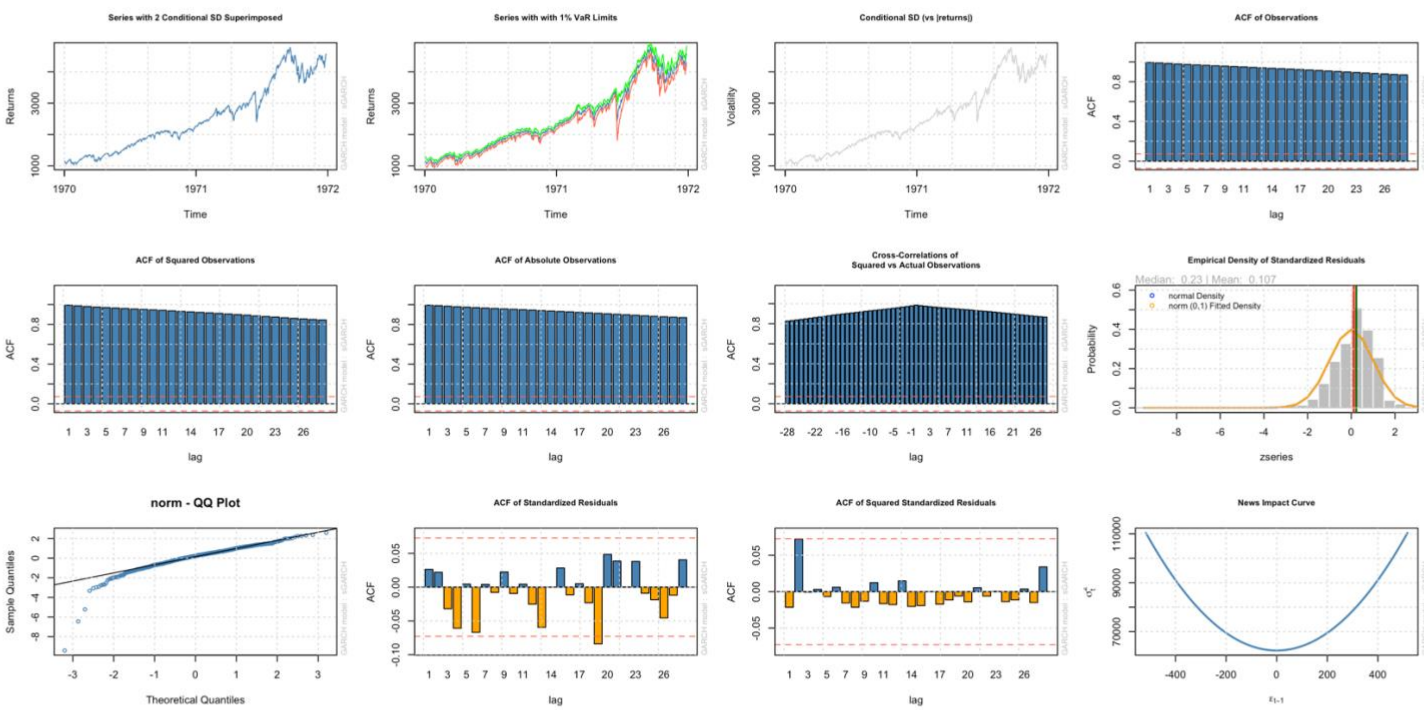


```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(2,0,1)
Distribution    : norm

Optimal Parameters
------------------------------------
         Estimate  Std. Error  t value Pr(>|t|)
mu     1151.78803   33.393144  34.4918 0.000000
ar1       0.47376    0.006706  70.6449 0.000000
ar2       0.52546    0.006718  78.2117 0.000000
ma1       0.40219    0.039721  10.1251 0.000000
omega    75.93658   43.227129   1.7567 0.078971
alpha1    0.18003    0.042615   4.2246 0.000024
beta1     0.81897    0.046686  17.5422 0.000000


Robust Standard Errors:
         Estimate  Std. Error  t value Pr(>|t|)
mu     1151.78803   12.123190  95.0070 0.000000
ar1       0.47376    0.004354 108.8205 0.000000
ar2       0.52546    0.004594 114.3829 0.000000
ma1       0.40219    0.038705  10.3910 0.000000
omega    75.93658  125.994525   0.6027 0.546710
alpha1    0.18003    0.086129   2.0902 0.036597
beta1     0.81897    0.130121   6.2939 0.000000

LogLikelihood : -3848.993

Information Criteria
------------------------------------

Akaike       10.623
Bayes        10.667
Shibata      10.622
Hannan-Quinn 10.640

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
```

```
                            statistic p-value
Lag[1]                          0.501  0.4791
Lag[2*(p+q)+(p+q)-1][8]         4.309  0.6065
Lag[4*(p+q)+(p+q)-1][14]        6.407  0.6763
d.o.f=3
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                           statistic p-value
Lag[1]                        0.3338  0.5634
Lag[2*(p+q)+(p+q)-1][5]       3.3853  0.3412
Lag[4*(p+q)+(p+q)-1][9]       3.8967  0.6064
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3] 0.0003072 0.500 2.000  0.9860
ARCH Lag[5] 0.0252363 1.440 1.667  0.9981
ARCH Lag[7] 0.1201029 2.315 1.543  0.9991

Nyblom stability test
------------------------------------
Joint Statistic:  2.4915
Individual Statistics:
mu     0.009837
ar1    0.046273
ar2    0.046299
ma1    0.054664
omega  0.616081
alpha1 0.272965
beta1  0.350092

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:          1.69 1.9 2.35
Individual Statistic:     0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           1.2913 0.1970
Negative Sign Bias  0.5795 0.5624
Positive Sign Bias  1.3591 0.1746
Joint Effect        6.9967 0.0720   *
```

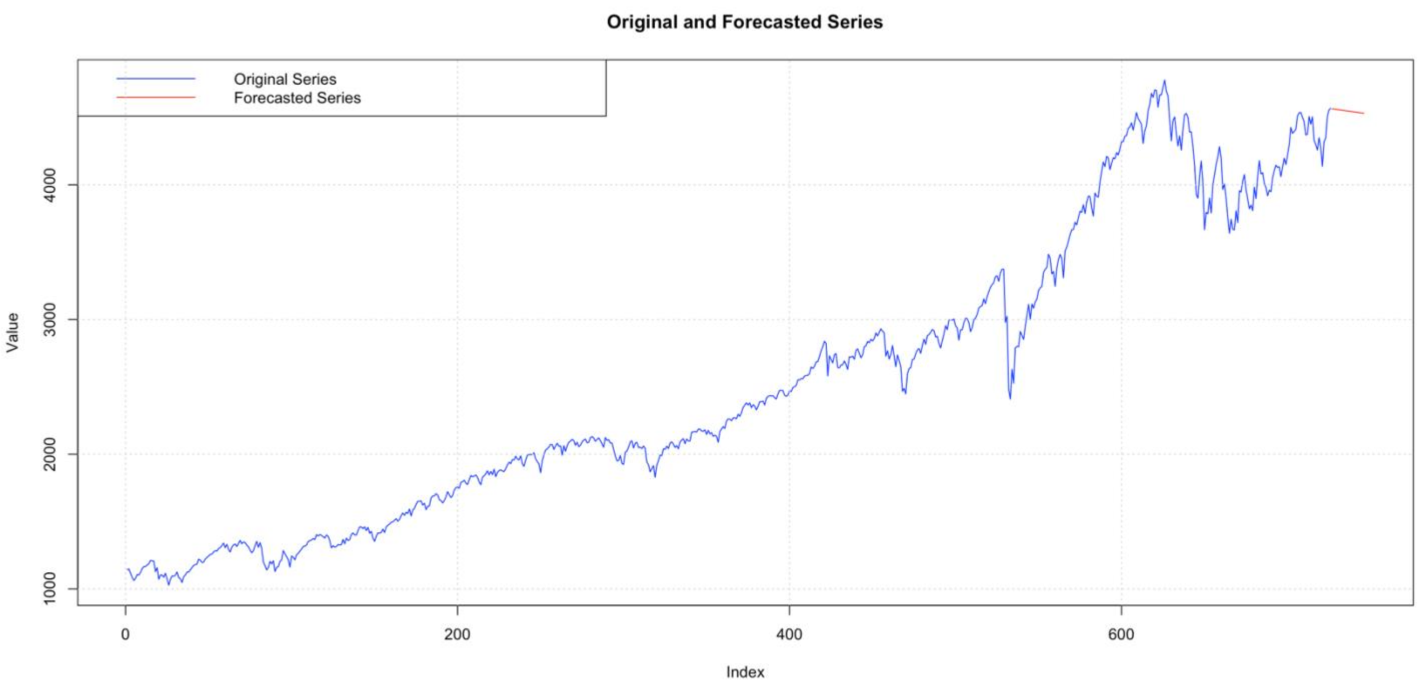
```
Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     75.60    1.054e-08
2    30     98.63    1.620e-09
3    40     91.08    4.746e-06
4    50    105.96    4.488e-06


Elapsed time : 0.340126
```

```
plot(garch2, which='all')
```




```
# Forecast the next 20 periods using GARCH
forecast1 <- ugarchforecast(fitORspec = garch2, n.ahead = 20)

# Plot the fitted values and forecasted values
plot(fitted(forecast1), type = 'l', col = 'blue', main = 'Fitted Values', ylab = 'Value')

# Plot conditional volatility (sigma)
```

```
plot(sigma(forecast1), type = 'l', col = 'blue', main = 'Conditional Volatility (sigma)', ylab =
'Volatility')

# Concatenate the original series and forecasted series
series <- c(temp_data_garch, rep(NA, length(fitted(forecast1))))
forecastseries <- c(rep(NA, length(temp_data_garch)), fitted(forecast1))

# Plot the original and forecasted series
par(mfrow = c(1, 1))
plot(series, type = "l", col = "blue", main = "Original and Forecasted Series", ylab = "Value")
lines(forecastseries, col = "red")
legend("topleft", legend = c("Original Series", "Forecasted Series"), col = c("blue", "red"), lty = 1)
# Add grid
grid()
```



Conclusion

This time series analysis project delved into Scallop sales spanning 26 years and the S&P500 index movement, employing advanced models like SARIMA, ARIMA, and GARCH. For Scallop sales, SARIMA(7,1,0)(5,0,0)[12] was selected based on AIC and BIC, passing diagnostic tests. The model accurately forecasted sales during the test period.

For the S&P500, ARIMA(2,1,4) achieved stationarity after differencing. Residual analysis validated model adequacy. Additionally, GARCH(1,1) captured volatility patterns. The combined approach showcased adaptability to alternative techniques.

Both datasets demonstrated forecasting potential for informed decision-making, emphasizing model robustness. The project contributes to time series understanding and highlights the versatility of model selection for unique datasets, guiding strategic decisions in fisheries and finance.

References

[1] Cryer, J. D., & Chan, K. S. (2008). Time series analysis: with applications in R (Vol. 2). NewYork: Springer

[2] Box, G. E. P., et al. (1994). Time series analysis: Forecasting and control.

[3] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and practice.

[4] Hamilton, J. D. (1994). Time series analysis (Vol. 2).

[5] Wei, W. W. S. (2006). Time series analysis: Univariate and multivariate methods.

[6] Brooks, C. (2008). Introductory econometrics for finance.

[7] Enders, W. (2014). Applied econometric time series.

[8] Tsay, R. S. (2010). Analysis of financial time series.

Appendix

[1] Scallop Sales Forecasting Code

```
# Loading libraries
library(zoo)

##
## Attaching package: 'zoo'

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

library(ggplot2)
library(forecast)

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

library(TSA)

## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar

library(tseries)

file_path <- "/Users/nayanikaranjan/Stevens/Fall_Semester_2023/MA641 Time Series/Sem
Project/scallops.csv"

# Read the CSV file into a data frame
scallop <- read.csv(file_path, header = TRUE)

# Convert the Date column to Date format
scallop$Date <- as.Date(scallop$Date, format = "%m/%d/%y")


# Create a time series object with frequency 12
scallop_ts <- ts(scallop$Sales, frequency = 12, start = c(1986, 1))
```

```
# Plot the time series
plot(scallop_ts, ylab = 'Scallop Sales',main = 'Scallop Sales Over Time', type = "o",
col = 'maroon', lwd = 2)
grid(col = 'gray', lty = 2)
legend("topright", legend = "Time Series Plot of Scallop Sales", col = "maroon", lwd
= 2, cex = 0.8)

# ACF and PACF plots for Scallops Sales
par(mfrow = c(2, 1))
acf(as.vector(scallop_ts), main = "ACF for Scallop Sales", lag.max =100)
pacf(as.vector(scallop_ts), main = "PACF for Scallop Sales", lag.max =100)

# ADF test to check for stationary
adf_result <- adf.test(scallop_ts)
print(adf_result)

##
##  Augmented Dickey-Fuller Test
##
## data:  scallop_ts
## Dickey-Fuller = -0.34663, Lag order = 6, p-value = 0.9885
## alternative hypothesis: stationary

# Calculate the first difference of the time series
scallop_diff <- diff(scallop_ts)

# ADF test to check for stationary
adf_result <- adf.test(scallop_diff)

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

print(adf_result)
##  Augmented Dickey-Fuller Test
## data:  scallop_diff
## Dickey-Fuller = -8.1578, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary

par(mfrow = c(1, 1))
plot(scallop_diff, ylab = 'First Difference of Scallop Sales',
     main = 'First Difference of Scallop Sales Over Time', type = "o",col = 'maroon',
lwd = 2)
grid(col = 'gray', lty = 2)
legend("topright", legend = "First Difference of Scallop Sales", col = "maroon", lwd
= 2, cex = 0.8)

# ACF and PACF plots for Scallops Sales
par(mfrow = c(2, 1))
acf(as.vector(scallop_diff), main = "ACF for First Difference Scallop Sales", lag.max
=100)
pacf(as.vector(scallop_diff), main = "PACF for First Difference Scallop Sales", lag.m
ax =100)
```

```
# Fit different SARIMA(p,d,q)(P,D,Q)S models
orders <- c(0, 1, 2, 3, 4, 6, 7)
seasonal_orders <- c(1, 2, 3, 4, 5)

best_model_aic <- NULL
best_model_bic <- NULL
best_aic_value <- Inf
best_bic_value <- Inf

for (p in orders) {
  for (P in seasonal_orders) {
    order <- c(p, 1, 0)
    seasonal_order <- c(P, 0, 0)
    sarima_model <- Arima(scallop_diff, order = order, seasonal = list(order = season
al_order, period = 12))

    aic_value <- AIC(sarima_model)
    bic_value <- BIC(sarima_model)

    # Check if the current model has lower AIC value
    if (aic_value < best_aic_value) {
      best_aic_value <- aic_value
      best_model_aic <- list(order = order, seasonal_order = seasonal_order, AIC = ai
c_value)
    }

    # Check if the current model has lower BIC value
    if (bic_value < best_bic_value) {
      best_bic_value <- bic_value
      best_model_bic <- list(order = order, seasonal_order = seasonal_order, BIC = bi
c_value)
    }
  }
}


if (identical(best_model_aic, best_model_bic)) {
  cat("Best Model (AIC/BIC):\n")
  print(best_model_aic)
  cat("\n")
} else {
  cat("Best Model (AIC):\n")
  print(best_model_aic)
  cat("\n")

  cat("Best Model (BIC):\n")
  print(best_model_bic)
  cat("\n")
}

## Best Model (AIC):
## $order
## [1] 7 1 0
##
```

```
## $seasonal_order
## [1] 5 0 0
##
## $AIC
## [1] 5409.936
##
##
## Best Model (BIC):
## $order
## [1] 6 1 0
##
## $seasonal_order
## [1] 5 0 0
##
## $BIC
## [1] 5456.98
```

```
# # Assuming best_model_aic is the best SARIMA model based on AIC
best_model_aic <- Arima(scallop_diff, order = c(7,1,0), seasonal = list(order = c(5,0
,0), period = 12))
# par(mfrow = c(1, 1))
# plot(best_model_aic)

set.seed(123)
par(mfrow = c(1, 1))
# Residual diagnostics
checkresiduals(best_model_aic)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(7,1,0)(5,0,0)[12]
## Q* = 20.301, df = 12, p-value = 0.0616
##
## Model df: 12.   Total lags used: 24
```

```
# Create a QQ plot for SARIMA(7,1,0)x12(5,0,0)
set.seed(123)
par(mfrow = c(1, 1))
residuals <- residuals(best_model_aic)

#Shapiro Test to check normality
print(shapiro.test(residuals))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.99359, p-value = 0.1873
```

```
# Assuming 'residuals' is your vector of residuals
plot(residuals, type='o', col='darkblue', pch=16, xlab='Date', ylab='Residuals', main
='Residuals Plot')
grid(col = 'darkblue', lty = 2
```

```
qqnorm(residuals)
qqline(residuals, col = 2)

# Reset the plotting layout
par(mfrow = c(1, 1))

# Ensure the time index is in order
scallop_diff <- ts(scallop_diff, start = c(1986, 1), frequency = 12)

# Define training set and test set
train_set <- window(scallop_diff, end = c(2009, 12))
test_set <- window(scallop_diff, start = c(2010, 1))

# ADF test to check for stationary
adf_result <- adf.test(test_set)

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

adf_result

##
##  Augmented Dickey-Fuller Test
##
## data:  test_set
## Dickey-Fuller = -5.4108, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary

# Fit SARIMA model on the training set
forecast_model <- Arima(train_set, order = c(7, 1, 0), seasonal = list(order = c(5, 1
, 0), period = 12))

# Forecast future values from 2009 to 2025
forecast_values <- forecast(forecast_model, h = 192)


# Plotting original and forecasted scallop sales
plot(scallop_diff, col = 'darkblue', type = 'o', lwd = 2, main = 'Original and Foreca
sted Scallop Sales')

# Adding color labels
legend('topright', legend = c('Original', 'Test Set', 'Forecasted'),
       col = c('darkblue', 'red', 'green'), lty = c(1, 2, 1), lwd = 2, bg = 'white')

# Plotting test set as a dotted line
lines(test_set, col = 'red', type = 'l', lwd = 2, lty = 2)

# Plotting forecasted values
lines(forecast_values$mean, col = 'green', type = 'o', lwd = 2)

# Adding grid lines
grid()
```

[2] S&P500 Forecasting Code

```
library(tseries)

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

library(TSA)

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar

library(forecast)

## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA

library(rugarch)

## Loading required package: parallel

##
## Attaching package: 'rugarch'

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

file_path <- "/Users/nayanikaranjan/Stevens/Fall_Semester_2023/MA641 Time Series/Sem
Project/sp500.csv"

# Read the CSV file into a data frame
sp500 <- read.csv(file_path, header = TRUE)

# Convert 'Date' column to Date format
sp500$Date <- as.Date(sp500$Date, format = "%Y-%m-%d")

# Check the structure and summary of the data
str(sp500)

## 'data.frame':    726 obs. of  7 variables:
##  $ Date     : Date, format: "2010-01-01" "2010-01-08" ...
##  $ Open     : num  1117 1141 1148 1115 1088 ...
##  $ High     : num  1142 1150 1150 1115 1105 ...
```

```
##  $ Low      : num  1117 1132 1115 1078 1063 ...
##  $ Close    : num  1142 1148 1116 1085 1063 ...
##  $ Adj.Close: num  1142 1148 1116 1085 1063 ...
##  $ Volume   : num  1.67e+10 2.14e+10 2.12e+10 2.62e+10 2.44e+10 ...
```

```
summary(sp500)
```

```
##       Date                 Open           High           Low      
##  Min.   :2010-01-01   Min.   :1028   Min.   :1071   Min.   :1011  
##  1st Qu.:2013-06-22   1st Qu.:1641   1st Qu.:1662   1st Qu.:1625  
##  Median :2016-12-12   Median :2251   Median :2273   Median :2213  
##  Mean   :2016-12-12   Mean   :2520   Mean   :2556   Mean   :2482  
##  3rd Qu.:2020-06-03   3rd Qu.:3234   3rd Qu.:3279   3rd Qu.:3211  
##  Max.   :2023-11-24   Max.   :4775   Max.   :4819   Max.   :4734  
##      Close        Adj.Close        Volume         
##  Min.   :1027   Min.   :1027   Min.   :5.038e+09  
##  1st Qu.:1650   1st Qu.:1650   1st Qu.:1.627e+10  
##  Median :2255   Median :2255   Median :1.860e+10  
##  Mean   :2524   Mean   :2524   Mean   :1.900e+10  
##  3rd Qu.:3246   3rd Qu.:3246   3rd Qu.:2.069e+10  
##  Max.   :4779   Max.   :4779   Max.   :4.123e+10
```

```
# Checking the "Date" column is in Date format
sp500Date <- as.Date(sp500$Date)
sp500Close <-  sp500$Close

# Plot the "Close" feature over time
plot(sp500Date, sp500Close, type = "l", col = "lightblue", lwd = 2,
     xlab = "Date", ylab = "Close Price", main = "S&P500 Close Price Over Time",
     sub = "From 1/1/10 to 11/30/23", cex.main = 1.2, cex.sub = 0.8)
grid()
legend("topleft", legend = "Close Price", col = "lightblue", lwd = 2, cex = 0.8)
```

```
# ACF and PACF plots for sp500Close
par(mfrow = c(2, 1))
acf(sp500Close, main = "ACF for S&P500Close", cex.main = 1.2, lag.max = 100)
pacf(sp500Close, main = "PACF for S&P500Close", cex.main = 1.2, lag.max =100)
```

```
# ADF test to check for stationary
adf_result <- adf.test(sp500Close)
print(adf_result)
```

```
## 
##  Augmented Dickey-Fuller Test
## 
## data:  sp500Close
## Dickey-Fuller = -2.4036, Lag order = 8, p-value = 0.4075
## alternative hypothesis: stationary
```

```
# ADF test on the first difference
sp500_close_diff <- diff(sp500Close)
adf_diff <- adf.test(sp500_close_diff)

## Warning in adf.test(sp500_close_diff): p-value smaller than printed p-value

print(adf_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  sp500_close_diff
## Dickey-Fuller = -10.16, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

# Plot differenced series
par(mfrow = c(1, 1))
plot(sp500Date[-1], sp500_close_diff, type = "o", col = "lightblue", lwd = 2,
     xlab = "Date", ylab = "y", main = "S&P500 Close After First Differencing", cex.m
ain = 1.2)
grid()
```

```
# ACF and PACF on the differenced series
par(mfrow = c(2, 1))
acf(sp500_close_diff, main = "ACF for S&P500Close", cex.main = 1.2, lag.max = 100)
pacf(sp500_close_diff, main = "PACF for S&P500Close", cex.main = 1.2, lag.max =100)
```

```
eacf(sp500_close_diff)

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

# Implementing ARIMA GARCH model
# Auto ARIMA
arima_model <- auto.arima(sp500Close)
arima_model

## Series: sp500Close
## ARIMA(2,1,4) with drift
##
## Coefficients:
##           ar1     ar2     ma1      ma2     ma3      ma4   drift
##       -0.1058  0.5798  0.0425  -0.5324  0.0304  -0.1622  4.6795
```

```
## s.e.   0.1256  0.1174  0.1270   0.1148  0.0403   0.0366  1.6457
##
## sigma^2 = 3810:  log likelihood = -4014.28
## AIC=8044.56   AICc=8044.76   BIC=8081.25
```

```
# Summary of the ARIMA model
summary(arima_model)
```

```
## Series: sp500Close
## ARIMA(2,1,4) with drift
##
## Coefficients:
##           ar1     ar2     ma1      ma2     ma3      ma4   drift
##       -0.1058  0.5798  0.0425  -0.5324  0.0304  -0.1622  4.6795
## s.e.   0.1256  0.1174  0.1270   0.1148  0.0403   0.0366  1.6457
##
## sigma^2 = 3810:  log likelihood = -4014.28
## AIC=8044.56   AICc=8044.76   BIC=8081.25
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.03436625 61.38728 39.47542 -0.07512502 1.549377 0.9758678
##                      ACF1
## Training set -0.001927647
```

```
# Print optimization output
print(arima_model$optim.output)
```

```
## NULL
```

```
# Manual ARIMA with specified order (2,1,4)
ns_model1 = arima(sp500Close, order=c(2,1,4))
print(ns_model1)
```

```
##
## Call:
## arima(x = sp500Close, order = c(2, 1, 4))
##
## Coefficients:
##           ar1     ar2     ma1      ma2     ma3      ma4
##       -0.1472  0.5463  0.0935  -0.4907  0.0365  -0.1538
## s.e.   0.1514  0.1399  0.1522   0.1354  0.0392   0.0361
##
## sigma^2 estimated as 3811:  log likelihood = -4017.84,  aic = 8047.68
```

```
# Extract nonseasonal residuals
nonseasonal_residuals_model1 = ns_model1$residuals

# Plotting residuals
par(mfrow = c(1, 1))
plot(nonseasonal_residuals_model1, type='o')
```

```
# ACF and PACF of residuals
par(mfrow = c(2, 1))
acf(nonseasonal_residuals_model1)
pacf(as.vector(nonseasonal_residuals_model1))
```

```
# QQ plot of residuals
par(mfrow = c(1, 1))
qqnorm(nonseasonal_residuals_model1)
qqline(nonseasonal_residuals_model1)
```

```
# Histogram of residuals
hist(nonseasonal_residuals_model1)
```

```
# Shapiro-Wilk normality test
print(shapiro.test(nonseasonal_residuals_model1))

##
##  Shapiro-Wilk normality test
##
## data:  nonseasonal_residuals_model1
## W = 0.86461, p-value < 2.2e-16

# Check residuals using checkresiduals function
checkresiduals(nonseasonal_residuals_model1)
```

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.6087, df = 10, p-value = 0.9633
##
## Model df: 0.   Total lags used: 10

# Reset plotting layout
par(mfrow = c(1, 1))

# Forecast using ARIMA model
model1 <- arima(ts(sp500Close, frequency = 12, start = c(2010, 1)), order = c(2, 1, 4
))
model1$x <- ts(sp500Close, frequency = 12, start = c(2010))
f = forecast(model1, level = c(95), h = 24)
plot(f)
```

```
# Fit GARCH model
temp_data_garch <- as.numeric(sp500Close)
```

```r
spec <- ugarchspec(
  variance.model = list(
    model = "sGARCH",
    garchOrder = c(1, 1),
    submodel = NULL,
    external.regressors = NULL,
    variance.targeting = FALSE
  ),
  mean.model = list(
    armaOrder = c(2, 1, 4),  # ARIMA(2,1,4)
    external.regressors = NULL
  ),
  distribution.model = "norm"
)
garch2 <- ugarchfit(spec = spec, data = temp_data_garch, solver.control = list(trace
= 0))

# Display GARCH model results
print(garch2)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(2,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error  t value Pr(>|t|)
## mu     1151.78803   33.393144  34.4918 0.000000
## ar1       0.47376    0.006706  70.6449 0.000000
## ar2       0.52546    0.006718  78.2117 0.000000
## ma1       0.40219    0.039721  10.1251 0.000000
## omega    75.93658   43.227129   1.7567 0.078971
## alpha1    0.18003    0.042615   4.2246 0.000024
## beta1     0.81897    0.046686  17.5422 0.000000
##
## Robust Standard Errors:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu     1151.78803   12.123190  95.0070 0.000000
## ar1       0.47376    0.004354 108.8205 0.000000
## ar2       0.52546    0.004594 114.3829 0.000000
## ma1       0.40219    0.038705  10.3910 0.000000
## omega    75.93658  125.994525   0.6027 0.546710
## alpha1    0.18003    0.086129   2.0902 0.036597
## beta1     0.81897    0.130121   6.2939 0.000000
##
## LogLikelihood : -3848.993
##
```

```
## Information Criteria
## ------------------------------------
##
## Akaike       10.623
## Bayes        10.667
## Shibata      10.622
## Hannan-Quinn 10.640
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       0.501  0.4791
## Lag[2*(p+q)+(p+q)-1][8]      4.309  0.6065
## Lag[4*(p+q)+(p+q)-1][14]     6.407  0.6763
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3338  0.5634
## Lag[2*(p+q)+(p+q)-1][5]    3.3853  0.3412
## Lag[4*(p+q)+(p+q)-1][9]    3.8967  0.6064
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3] 0.0003072 0.500 2.000  0.9860
## ARCH Lag[5] 0.0252363 1.440 1.667  0.9981
## ARCH Lag[7] 0.1201029 2.315 1.543  0.9991
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.4915
## Individual Statistics:
## mu     0.009837
## ar1    0.046273
## ar2    0.046299
## ma1    0.054664
## omega  0.616081
## alpha1 0.272965
## beta1  0.350092
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.2913 0.1970
## Negative Sign Bias  0.5795 0.5624
```

```
## Positive Sign Bias  1.3591 0.1746
## Joint Effect        6.9967 0.0720   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     75.60    1.054e-08
## 2    30     98.63    1.620e-09
## 3    40     91.08    4.746e-06
## 4    50    105.96    4.488e-06
##
##
## Elapsed time : 0.1425402
```

```
plot(garch2, which='all')
```

```
##
## please wait...calculating quantiles...
```

```
# Forecast the next 20 periods using GARCH
forecast1 <- ugarchforecast(fitORspec = garch2, n.ahead = 20)

# Plot the fitted values and forecasted values
plot(fitted(forecast1), type = 'l', col = 'blue', main = 'Fitted Values', ylab = 'Val
ue')
```

```
# Plot conditional volatility (sigma)
plot(sigma(forecast1), type = 'l', col = 'blue', main = 'Conditional Volatility (sigm
a)', ylab = 'Volatility')
```

```
# Concatenate the original series and forecasted series
series <- c(temp_data_garch, rep(NA, length(fitted(forecast1))))
forecastseries <- c(rep(NA, length(temp_data_garch)), fitted(forecast1))

# Plot the original and forecasted series
par(mfrow = c(1, 1))
plot(series, type = "l", col = "lightblue", main = "Original and Forecasted Series",
ylab = "Value")
lines(forecastseries, col = "darkblue")
lines(series, col = "lightblue", lty = 2)  # Dotted line for the original series
legend("topleft", legend = c("Original Series", "Forecasted Series"), col = c("lightb
lue", "darkblue"), lty = 1)

# Add grid
grid()
```