Seasonal and Nonseasonal GARCH Time Series Analysis

Jaykumar Patel

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

Project Supervisor: Dr. Hadi Safari Katesari

PART A: SEASONAL DATASET: perrin freres monthly champagne sales data

Introduction and Motivation

The motivation for analyzing this dataset is to gain insights into the trends, seasonality, and other patterns in champagne sales over time. This information can be useful for Perrin Freres to forecast future sales, make informed business decisions, and identify areas for improvement in their sales strategies.

In addition, analyzing this dataset can provide valuable insights for other businesses in the food and beverage industry that rely on seasonal sales patterns. By understanding the trends and seasonality in sales, businesses can make better decisions regarding inventory management, pricing strategies, and marketing campaigns.

Overall, the Perrin Freres monthly champagne sales dataset presents an interesting and challenging time series analysis project that can provide valuable insights for businesses and researchers alike.

Data Description

Date Range: From Jan 1964 to sept 1972

Datasource Description: The data is from Kaggle website and can be accessed using the link: https://www.kaggle.com/datasets/anupamshah/perrin-freres-monthly-champagne-sales (https://www.kaggle.com/datasets/anupamshah/perrin-freres-monthly-champagne-sales).

Dataset Description: The dataset contains 105 entries, 2 total columns. One is date and another is sales.

```
library(TSA)
```

```
## Warning: package 'TSA' was built under R version 4.2.2
```

```
##
## Attaching package: 'TSA'
```

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

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

```
data <- read.csv("perrin-freres-monthly-champagne.csv")
```

```
summary(data)
```

```
##     Month               sales
##  Length:107         Min.   : 1413
##  Class :character   1st Qu.: 3113
##  Mode  :character   Median : 4217
##                     Mean   : 4761
##                     3rd Qu.: 5221
##                     Max.   :13916
##                     NA's   :2
```

Checking seasonality

```
library(seastests)
```

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

```
isSeasonal(data$sales, test = "combined", freq = 12)
```

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

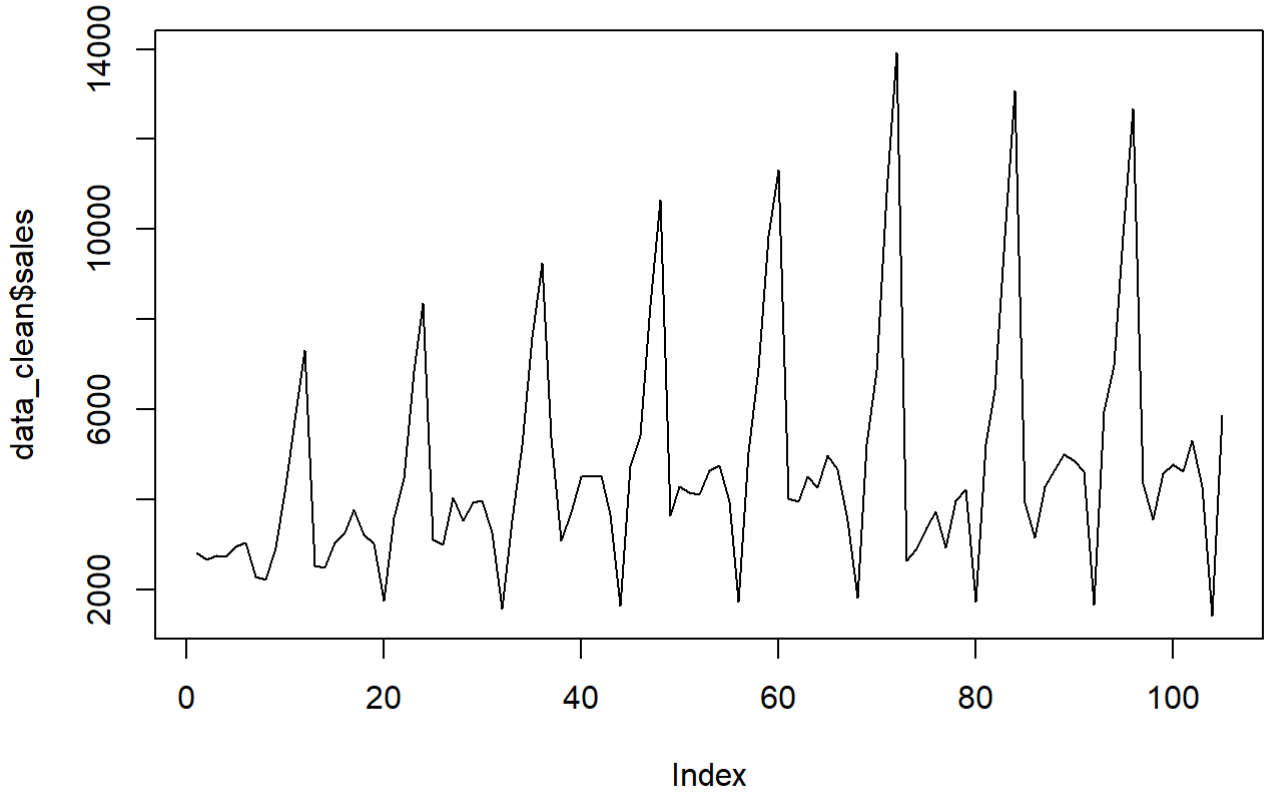
```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

```
## [1] TRUE
```

Data Pre-processing

Before any further preprocessing we want to remove the null values.

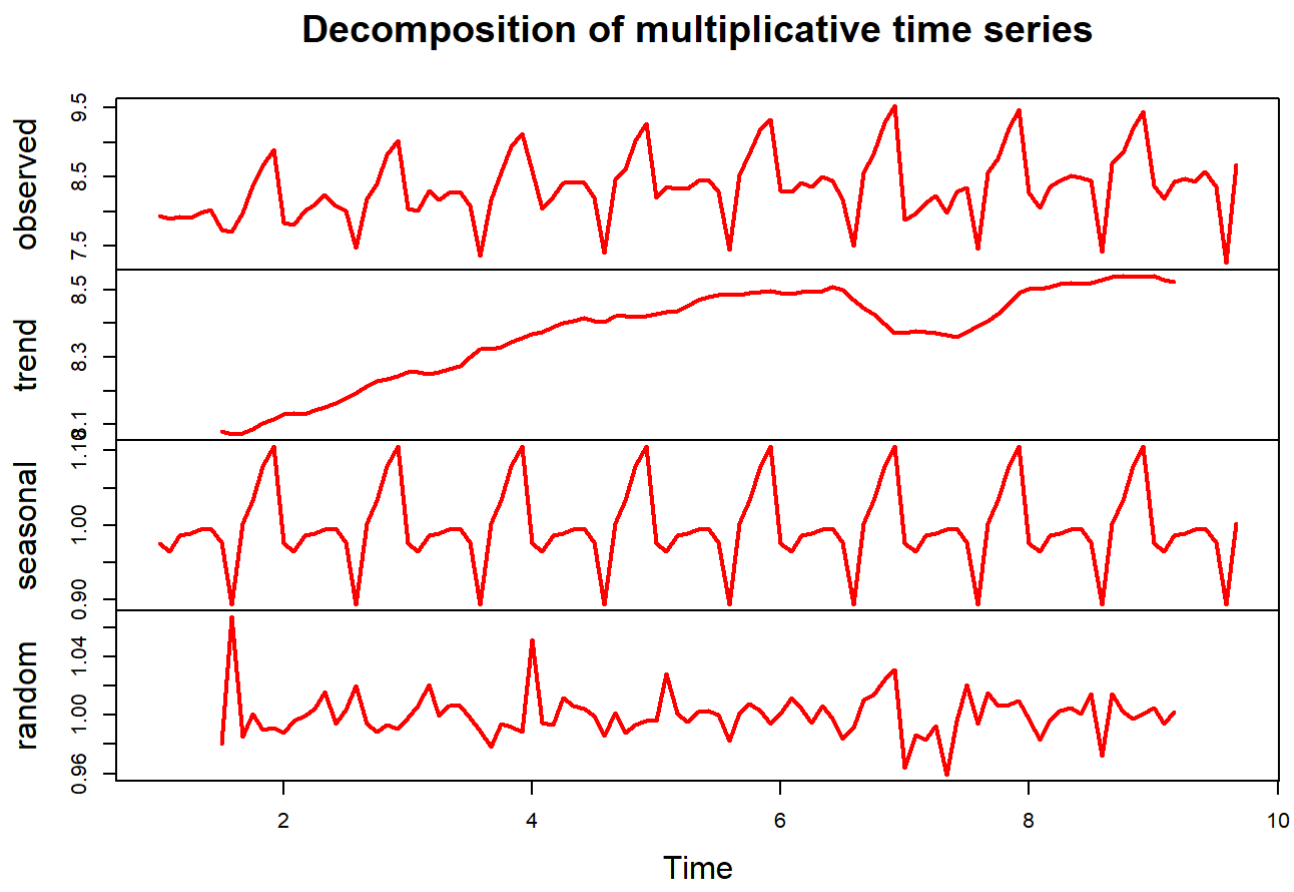
```
data_clean <- na.omit(data)
plot(data_clean$sales, type='l')
```



Decomposing the time series

Decomposing the time series to have a look at the seasonal components, trend components, and residuals in it.

```
ts_sales = ts(log(data_clean$sales), frequency = 12)
decompose_sales = decompose(ts_sales, "multiplicative")
plot(decompose_sales, type='l',lwd=2, col = 'red')
```



The series exhibits multiplicative decomposition. As the amplitude of both the seasonal and irregular variations increase as the level of the trend rises. In the multiplicative model, the original time series is expressed as the product of trend, seasonal and irregular components.

Stationarity check

Let's start by checking if the time series is stationary or not. To do so we are going to use the Dickey Fuller and/or augmented Dickey fuller test

```
library(aTSA)
```

```
##
## Attaching package: 'aTSA'
```

```
## The following object is masked from 'package:graphics':
##
##     identify
```

```
adf.test(data_clean$sales)
```

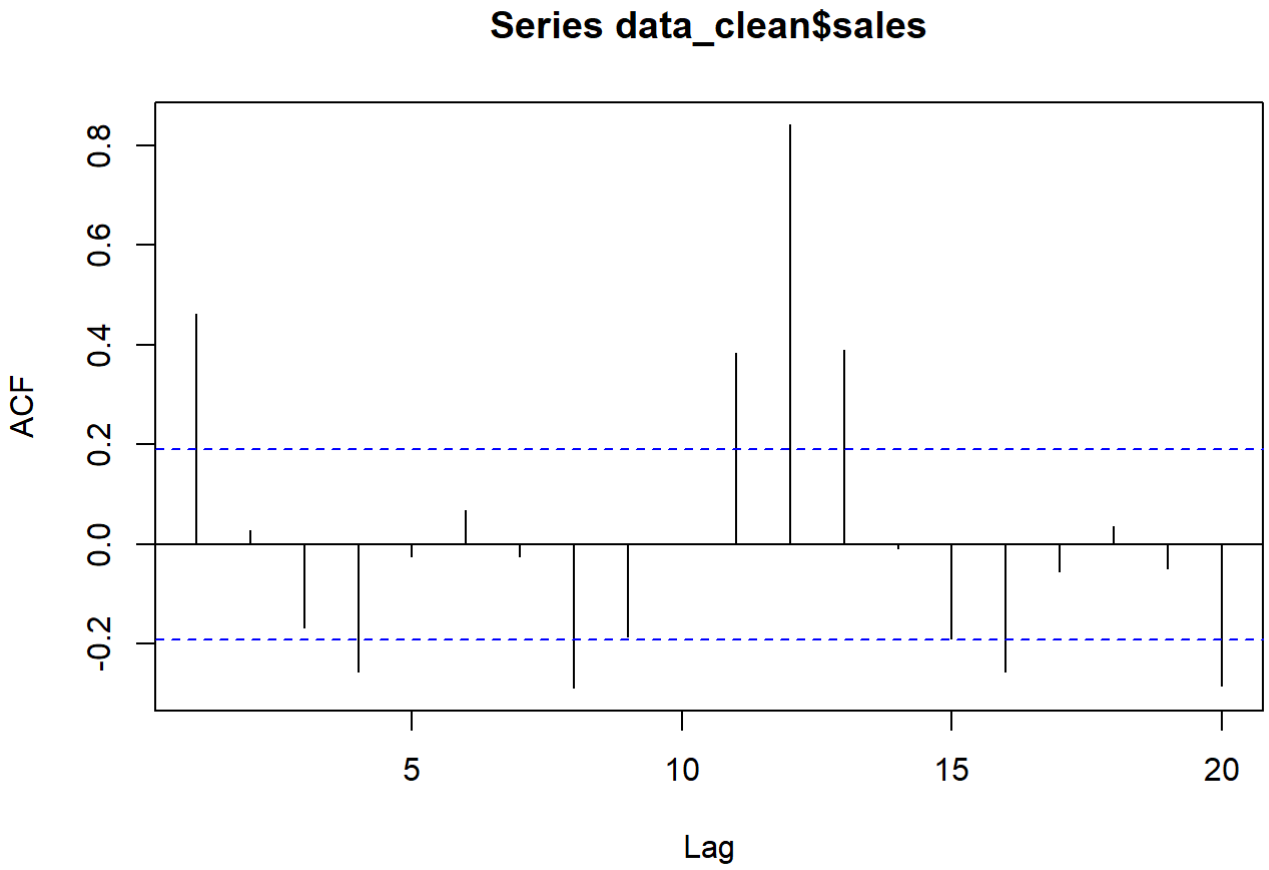
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -2.461  0.0158
## [2,]   1 -2.280  0.0233
## [3,]   2 -1.769  0.0767
## [4,]   3 -1.490  0.1435
## [5,]   4 -0.863  0.3691
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -6.13    0.01
## [2,]   1 -6.69    0.01
## [3,]   2 -6.09    0.01
## [4,]   3 -6.09    0.01
## [5,]   4 -4.25    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -6.39    0.01
## [2,]   1 -7.15    0.01
## [3,]   2 -6.69    0.01
## [4,]   3 -6.96    0.01
## [5,]   4 -4.89    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The augmented Dickey Fuller test demonstrates that the P values are not less than.05. Therefore, the series is not stationary, and the null hypothesis must be rejected. In other words, the variance is not constant over time and the time series has some sort of time dependent structure.

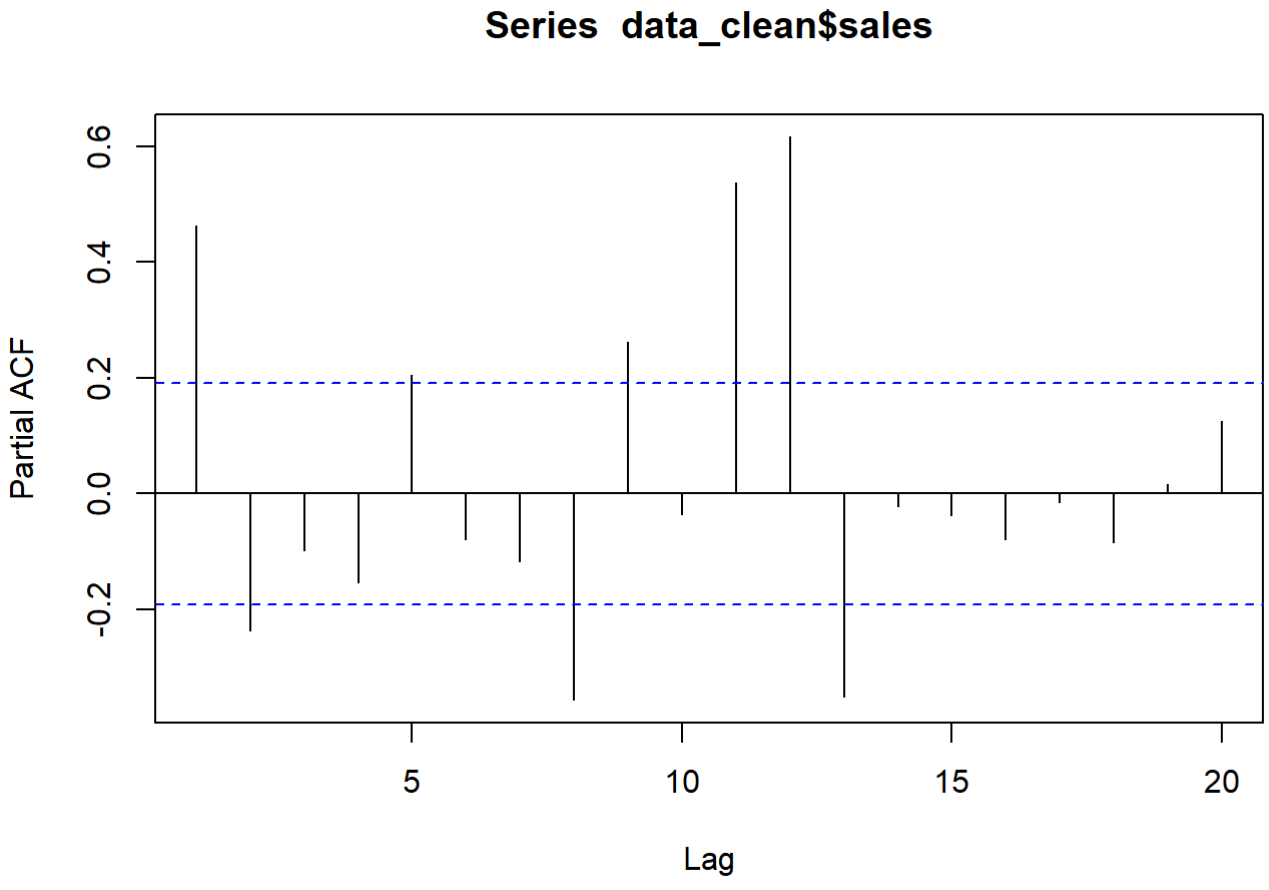
Before fitting a model to the series, it is crucial to make it stationary because we only ever see one instance of a stochastic process, as opposed to many instances. So, in order for watching a lengthy run of a stochastic process to be comparable to observing numerous independent runs of a stochastic process, stationarity and ergodicity are required.

Plotting the ACF and PCF of the series.

```
acf(data_clean$sales)
```




```
pacf(data_clean$sales)
```



The data's autocorrelation plot shows relatively little deterioration. This supports the finding that the time series is not stationary from the Dickey-Fuller test. A strong autocorrelation is apparent from the trend in the Data, which is visible.

MAKING THE TIME SERIES STATIONARY

Using the Box Cox transformation:

A parameter lambda is used to index the Box-Cox transformation family of power transformations. Anytime we have a non-stationary time series (with non-constant variance), we can utilize this transformation. When Box-Cox is used with a specific lambda value, the process could become stationary.

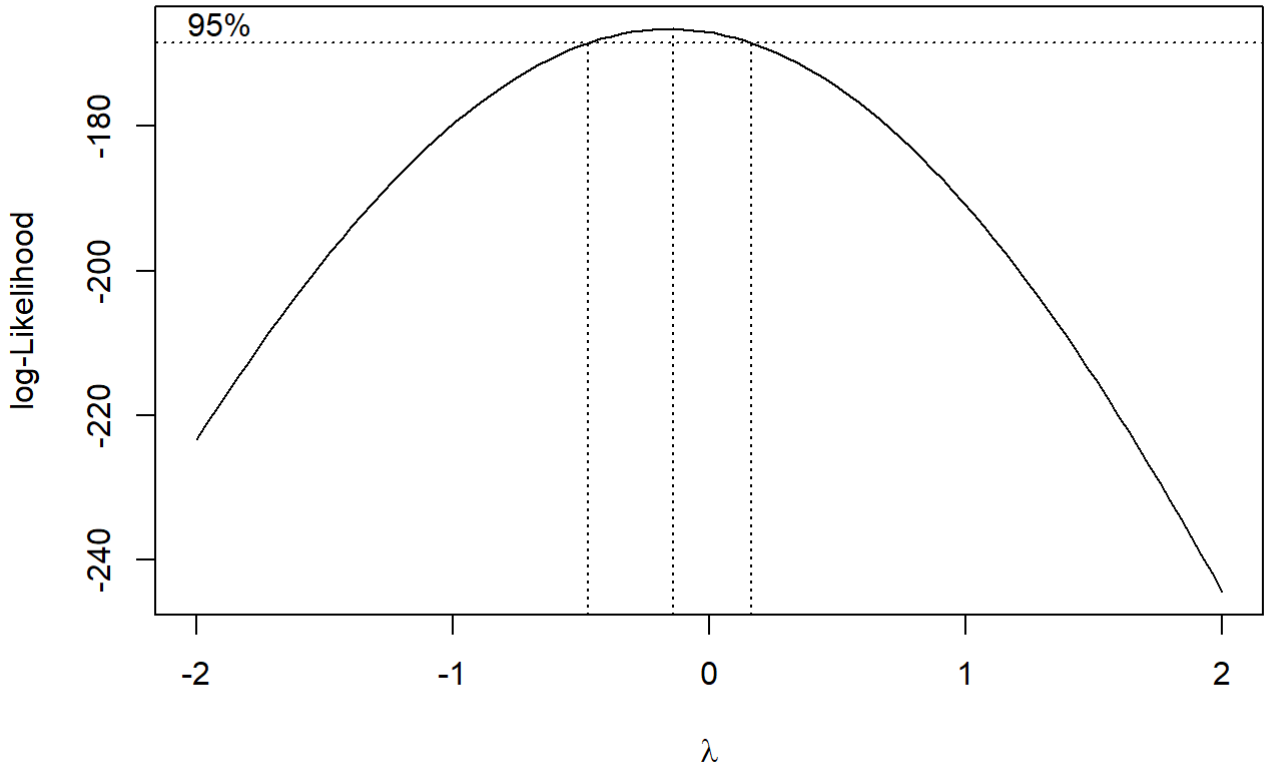
```
library(MASS)
library(forecast)
```

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

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

```
## The following object is masked from 'package:aTSA':
##
##     forecast
```

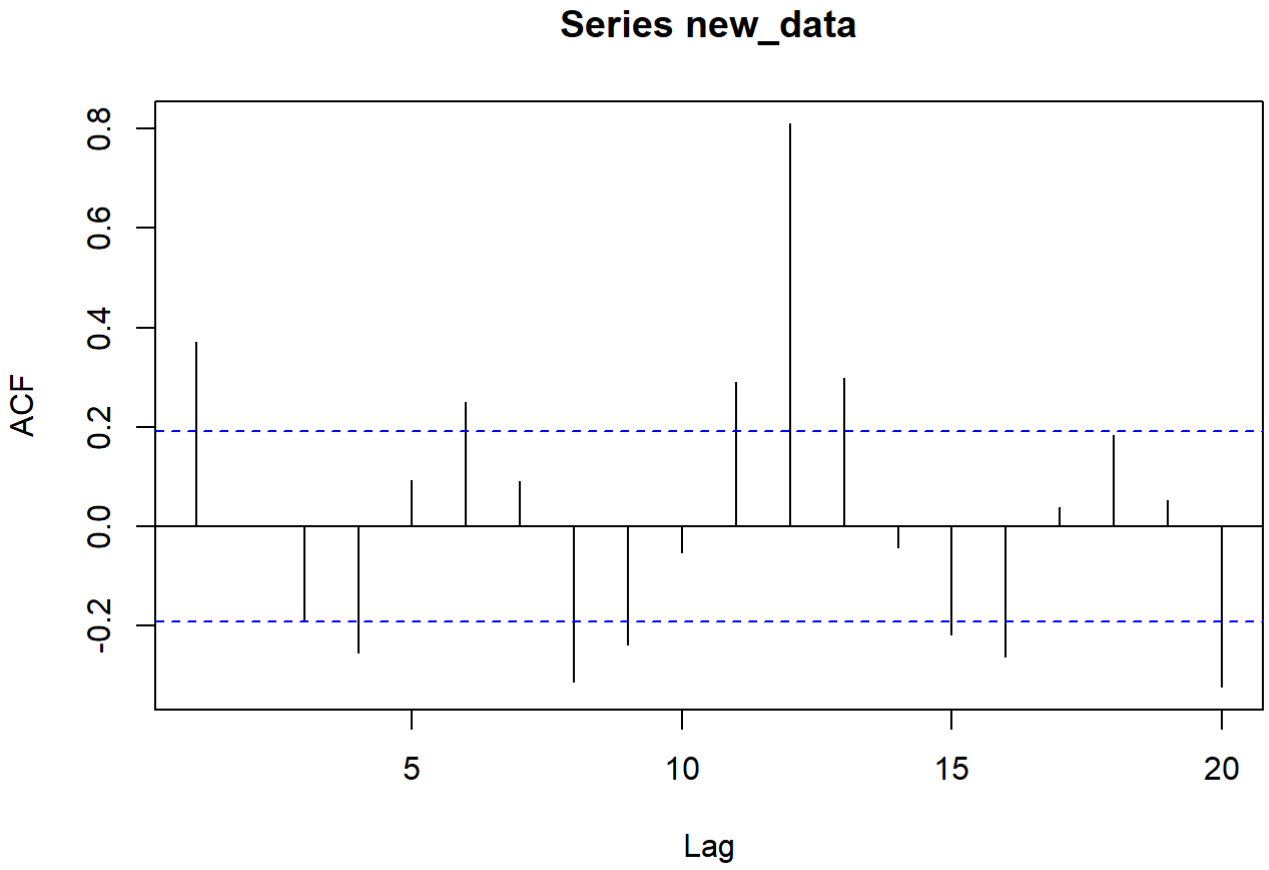
```
b <- boxcox(lm(data_clean$sales ~ 1))
```



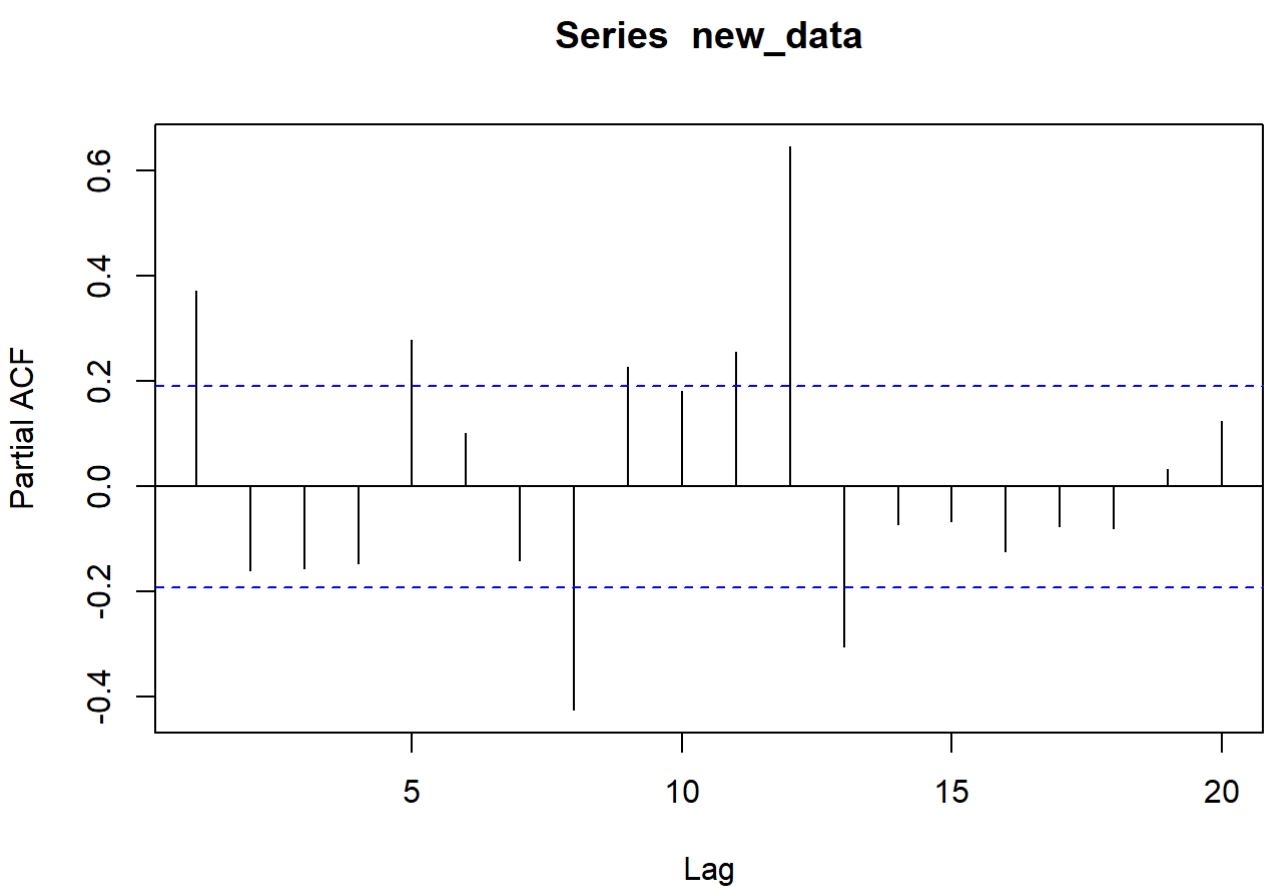
```
lambda <- b$x[which.max(b$y)]
lambda
```

```
## [1] -0.1414141
```

```
new_data <- (data_clean$sales ^ lambda - 1) / lambda
acf(new_data)
```




```
pacf(new_data)
```




```
adf.test(new_data)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag     ADF p.value
## [1,]   0 -0.0353   0.633
## [2,]   1 -0.0208   0.637
## [3,]   2  0.0171   0.648
## [4,]   3  0.0652   0.662
## [5,]   4  0.1845   0.696
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -6.83    0.01
## [2,]   1 -6.72    0.01
## [3,]   2 -6.49    0.01
## [4,]   3 -6.34    0.01
## [5,]   4 -4.04    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -7.14    0.01
## [2,]   1 -7.21    0.01
## [3,]   2 -7.20    0.01
## [4,]   3 -7.39    0.01
## [5,]   4 -4.66    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

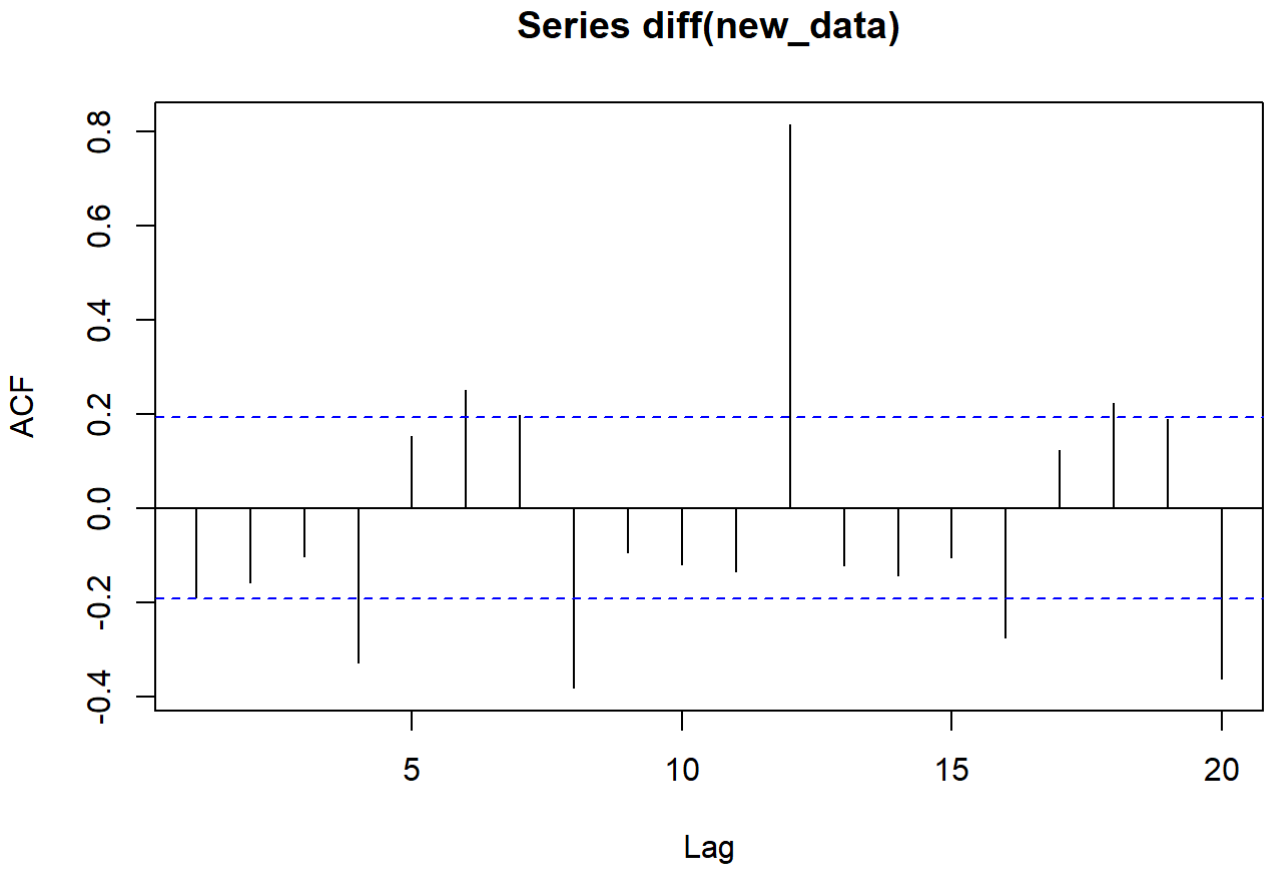
We can observe that the series is still moving even after the Box Cox change. The series is not stationary since the P value is bigger than 0.5. Additionally, the acf and pacf plots demonstrate that the time series still exhibits autocorrelation. Let's do a seasonal differentiation of the time series to eliminate this association and make the series stationary.

SEASONAL DIFFERENCING

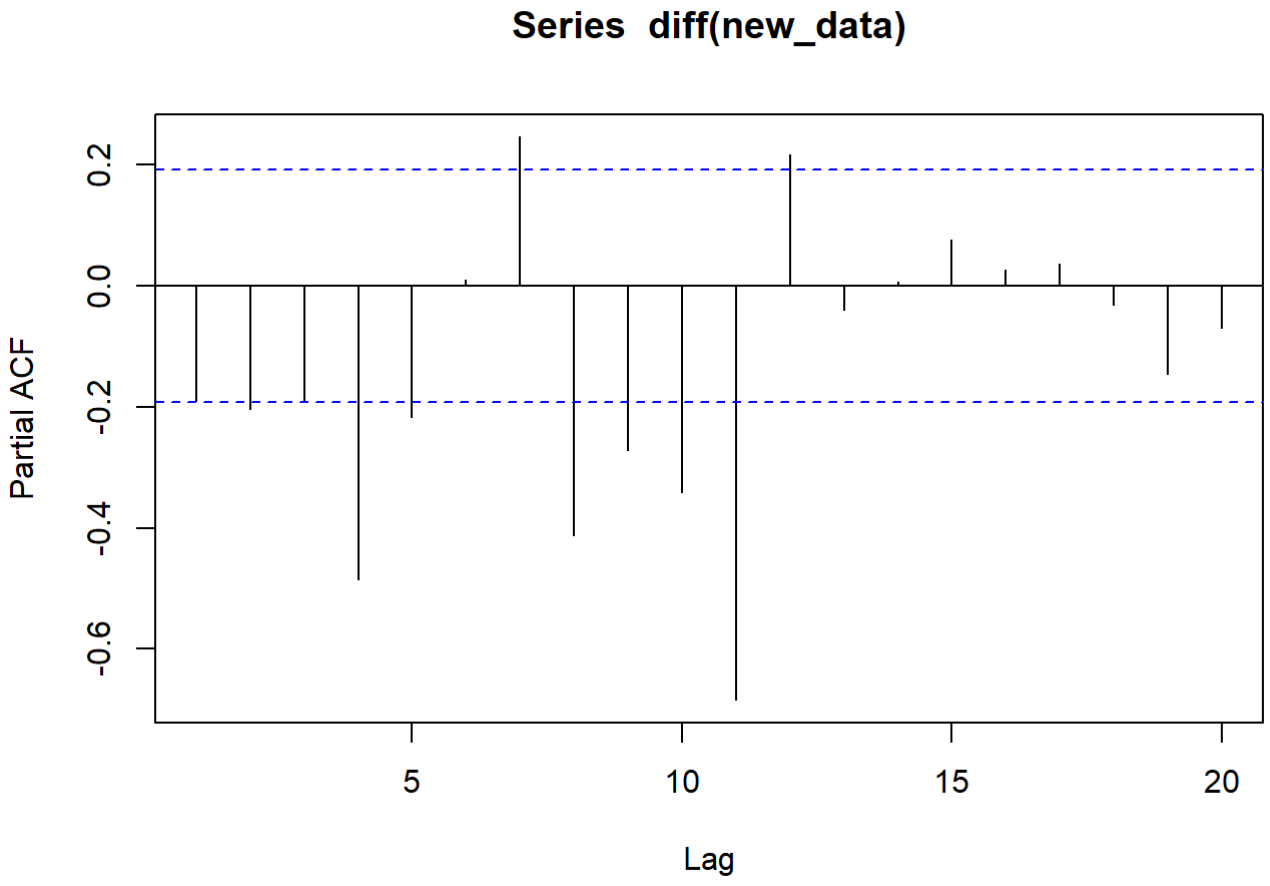
```
diff_ser <- diff(new_data)
adf.test(diff(new_data,lag = 12))
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -6.98    0.01
## [2,]   1 -4.50    0.01
## [3,]   2 -3.24    0.01
## [4,]   3 -2.71    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -7.52  0.0100
## [2,]   1 -4.97  0.0100
## [3,]   2 -3.58  0.0100
## [4,]   3 -2.96  0.0448
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -7.77  0.0100
## [2,]   1 -5.26  0.0100
## [3,]   2 -3.86  0.0195
## [4,]   3 -3.19  0.0943
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
acf(diff(new_data))
```




```
pacf(diff(new_data))
```



```
eacf(diff(new_data))
```

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

After comparing the series, we can observe that the P value from the enhanced Dickey-Fuller test is less than 0.05, which indicates that it is statistically significant. As a result, we can say that the series is stationary at this point.Additionally, we can see that the correlations have greatly decreased in the ACF and PCF lots.

Now that we have a stationary series, we'll have a look at the EACF plots to finalize the order of our AR, MA models and then fit the model.

DETERMINING THE ORDER OF THE MODEL

The best model to fit the data can have p, q, P, and Q in the range of 0 to 3, according to the ACF, PACF, and EACF. Based on the lowest AIC for P, Q, p, and q values, we will choose the best model.

I've created a nested for loop that goes through each conceivable combination of P, Q, p, and q values in the range of 0 to 3 and fits a SARIMA model for each of them. The AICs for each of these models are then listed along with the matching P, Q, p, and q values. The top value in the list is then popped once the list has been sorted in ascending order. The P, Q, p, and q values that correspond to the lowest AIC model are represented by this value.

NESTED FOR LOOP FOR DERTMING THE VALUES OF P, Q, p and q

```
# Load the forecast package
library(forecast)

# Define the range of values for p, q, P, and Q
p_values <- c(0, 1, 3)
q_values <- c(0, 1, 3)
P_values <- c(0, 1, 3)
Q_values <- c(0, 1, 3)

# Initialize variables for storing the best model and its performance
best_model <- NULL
best_aic <- Inf

# Nested for loops to iterate over different parameter values
for (p in p_values) {
  for (q in q_values) {
    for (P in P_values) {
      for (Q in Q_values) {

        # Fit a seasonal ARIMA model with the current parameter values
        fit <- arima(data_clean$sales, order=c(p,1,q), seasonal=c(P,1,Q), method="ML")

        # Evaluate the model performance using AIC
        current_aic <- AIC(fit)

        # Update the best model and its performance if the current model is better
        if (current_aic < best_aic) {
          best_model <- fit
          best_aic <- current_aic
          best_params <- c(p, q, P, Q)
        }
      }
    }
  }
}
```

```
## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced
```

```
# Print the best SARIMA model parameters and AIC
print(paste("Best SARIMA model parameters:", paste(best_params, collapse=",")))
```

```
## [1] "Best SARIMA model parameters: 3,3,1,3"
```

```
print(paste("Best AIC:", best_aic))
```

```
## [1] "Best AIC: 1869.95463051615"
```

Parameter Estimation using best model

```
(fit <- arima(data_clean$sales, order = c(3,1,3)))
```

```
##
## Call:
## arima(x = data_clean$sales, order = c(3, 1, 3))
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3
##       0.5630  0.3081  -0.3571  -1.0642  -0.7230  0.8300
## s.e.  0.1231  0.1512   0.1190   0.1015   0.1234  0.0828
##
## sigma^2 estimated as 3791574:  log likelihood = -938.05,  aic = 1888.09
```

```
(fit2 <- arima(data_clean$sales, order = c(3,1,5)))
```

```
##
## Call:
## arima(x = data_clean$sales, order = c(3, 1, 5))
##
## Coefficients:
##          ar1      ar2     ar3      ma1      ma2      ma3      ma4     ma5
##       0.2645  -0.3000  0.3042  -0.7903  -0.1345  -0.4724  -0.2901  0.7649
## s.e.  0.1465   0.1879  0.1207   0.1223   0.1956   0.1150   0.1171  0.0930
##
## sigma^2 estimated as 3341545:  log likelihood = -933.38,  aic = 1882.77
```

As per the aic I got the best model (3,1,3) but due to showing dependancies in Ljung-box test I decided to go with Arima(3,1,5) which is suggested by eacf and it works better.

```
# Load the "forecast" package
library(forecast)

# Fit an ARIMA model to the "sales" dataset
arima_model <- Arima(data_clean$sales, order = c(3,1,5))

# Make a seasonal ARIMA (SARIMA) model from the ARIMA model
sarima_model <- Arima(data_clean$sales, order = c(3,1,5), seasonal = list(order = c(1,1,3), p
eriod = 12))
# Print the model summaries
summary(arima_model)
```

```
## Series: data_clean$sales
## ARIMA(3,1,5)
##
## Coefficients:
##          ar1      ar2     ar3      ma1      ma2      ma3      ma4     ma5
##       0.2645  -0.3000  0.3042  -0.7903  -0.1345  -0.4724  -0.2901  0.7649
## s.e.  0.1465   0.1879  0.1207   0.1223   0.1956   0.1150   0.1171  0.0930
##
## sigma^2 = 3620008:  log likelihood = -933.38
## AIC=1884.77   AICc=1886.68   BIC=1908.57
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 255.4961 1819.264 1399.104 -9.843906 37.67181 0.8200434
##                      ACF1
## Training set -0.008447569
```

```
summary(sarima_model)
```

```
## Series: data_clean$sales
## ARIMA(3,1,5)(1,1,3)[12]
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2     ma3      ma4      ma5
##       0.0026  -0.4307  -0.4428  -0.7208  0.3613  0.1956  -0.5210  -0.2153
## s.e.  2.0069   1.2372   1.5789   1.9995  2.6889  2.3434   0.9432   0.5913
##          sar1     sma1    sma2     sma3
##       -0.2411  -0.0472  0.0431  -0.1117
## s.e.   1.2932   1.3002  0.4442   0.2185
##
## sigma^2 = 507648:  log likelihood = -732.35
## AIC=1490.69   AICc=1495.36   BIC=1523.47
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE
## Training set -20.90154 621.916 429.2668 -3.348272 11.02379 0.2516019
##                     ACF1
## Training set -0.01147262
```

These summaries provide information about the model coefficients, standard errors, and other statistics. By examining these summaries, we can assess the goodness of fit of the models and evaluate their forecasting performance.

Residual Analysis

```
# Load the "forecast" package
library(forecast)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

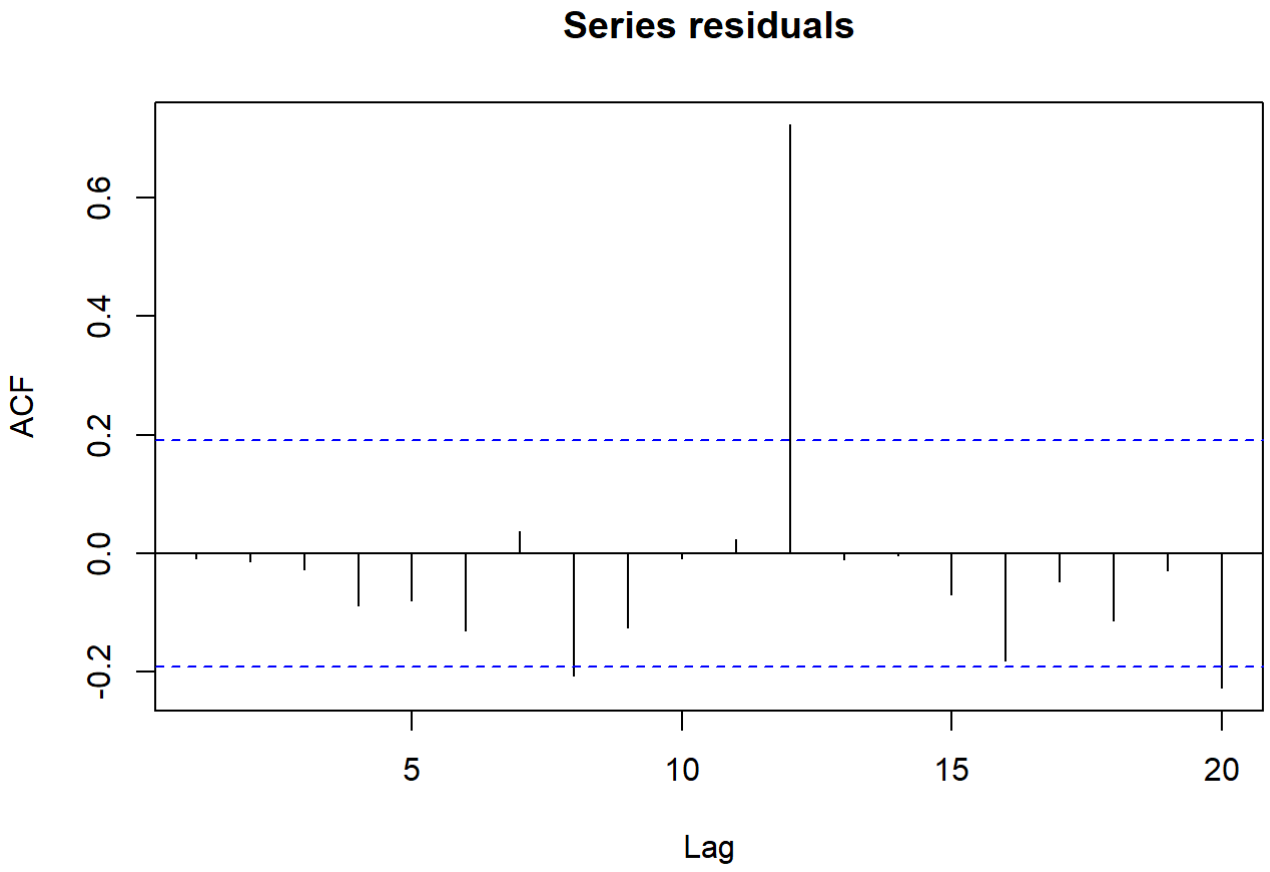
```
library(car)
```

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

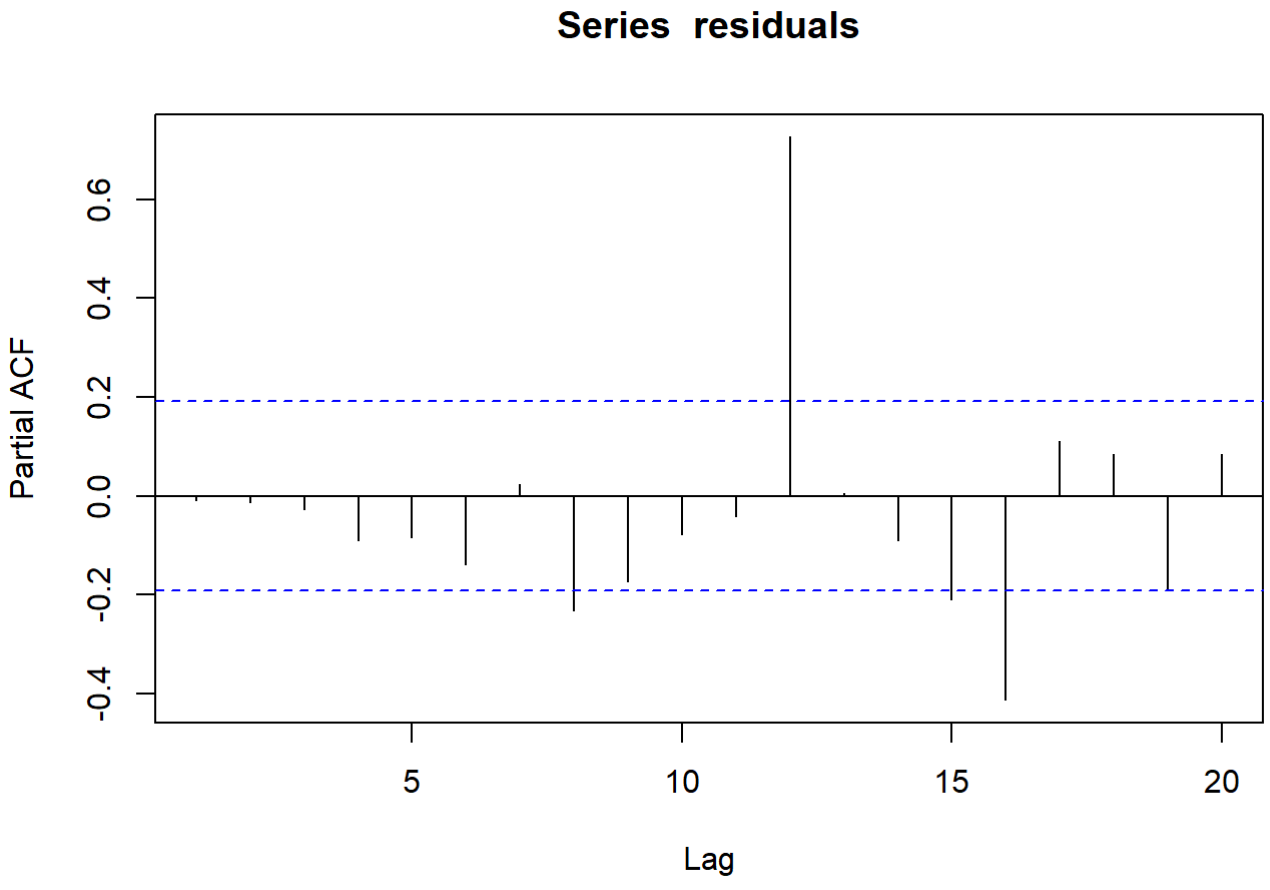
```
## Loading required package: carData
```

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

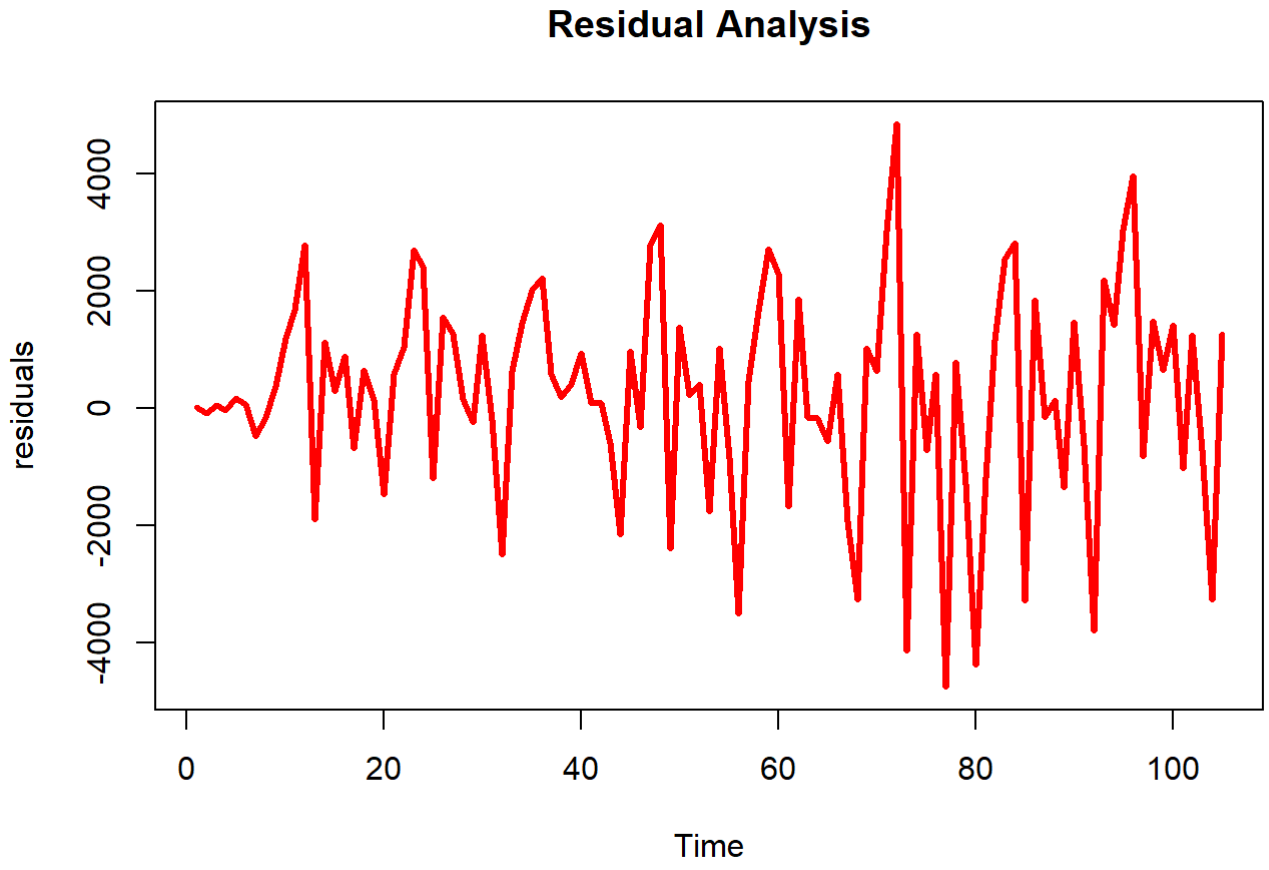
```
# Fit an ARIMA(3,1,3) model to the time series
arima_model <- Arima(data_clean$sales, order = c(3,1,5))

# Extract the residuals from the ARIMA model
residuals <- residuals(arima_model)

# Plot the ACF and PACF of the residuals
acf(residuals)
```



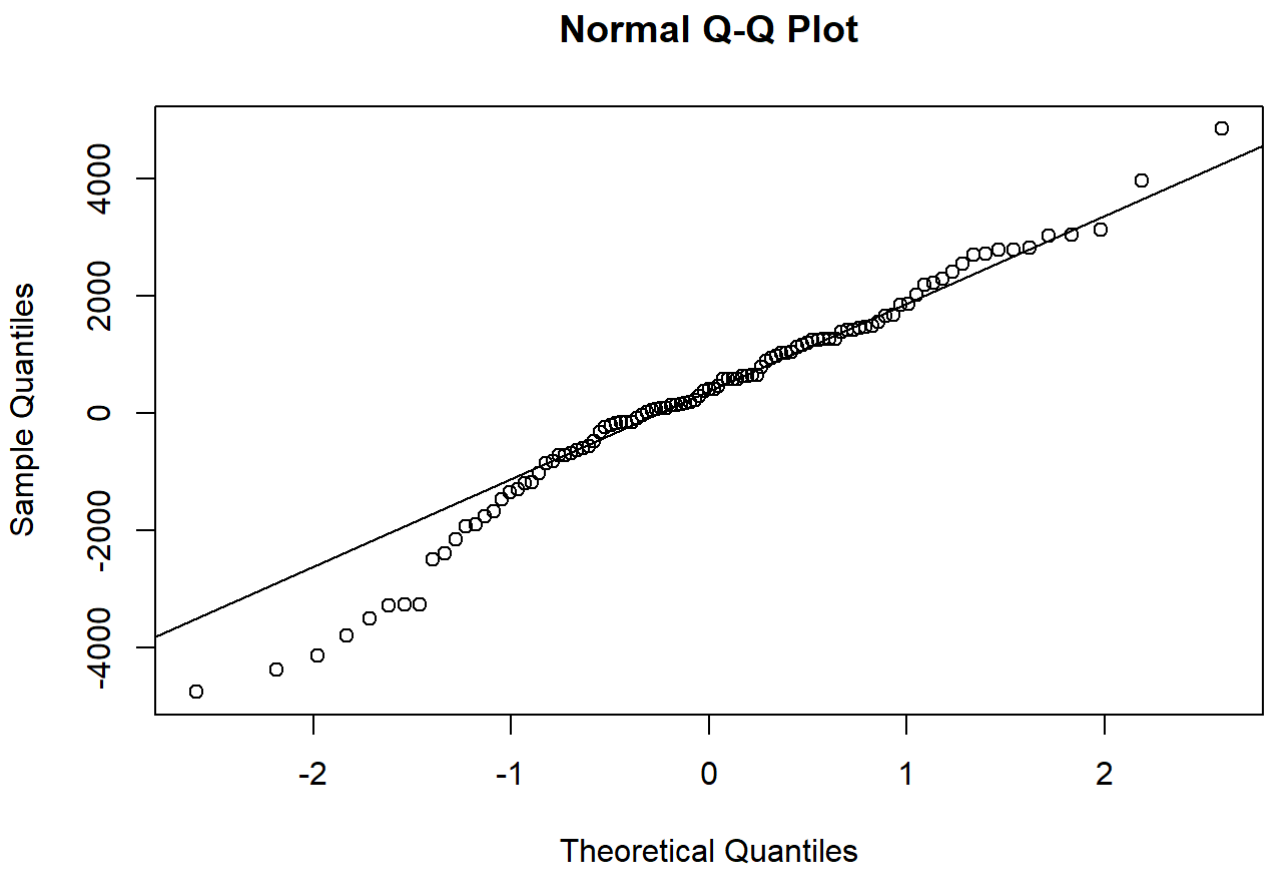

```
pacf(residuals)
```



```
#plot time series data
ts.plot(residuals,lwd=3,col="red",main='Residual Analysis')
```

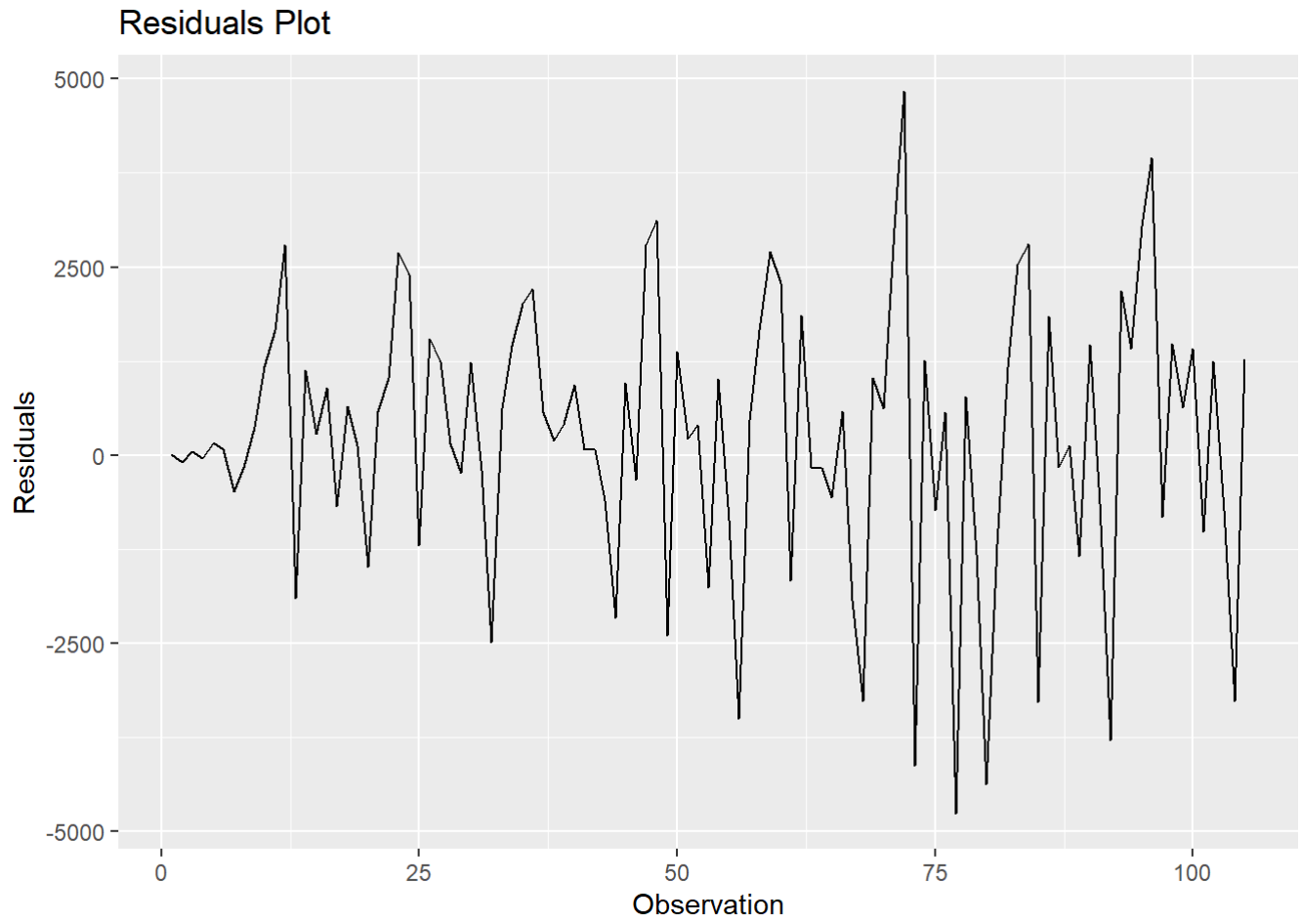



```
qqnorm(residuals)
qqline(residuals)
```

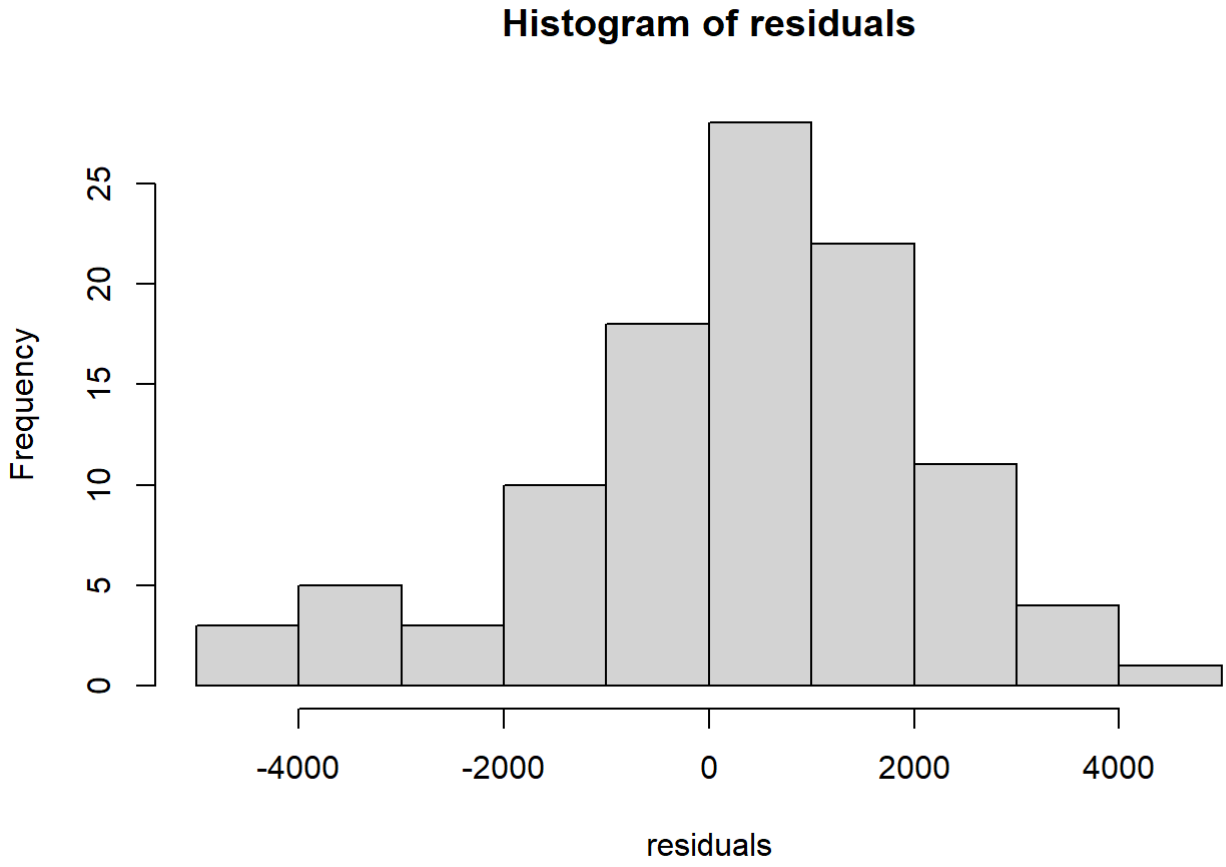



```
ggplot(data.frame(residuals = residuals), aes(x = 1:length(residuals), y = residuals)) +
  geom_line() +
  labs(x = "Observation", y = "Residuals", title = "Residuals Plot")
```

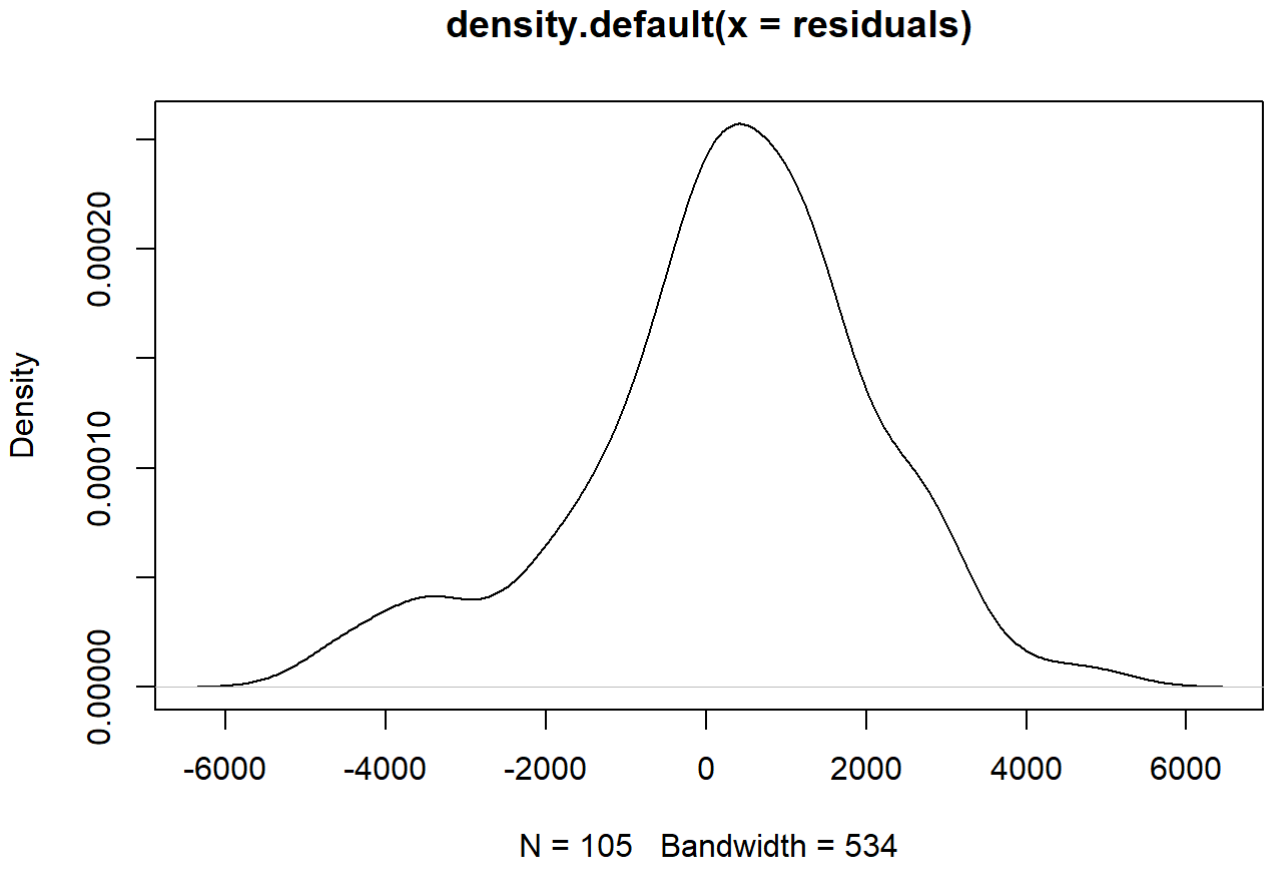
```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```




```
# Plot the histogram and density of the residuals
hist(residuals)
```



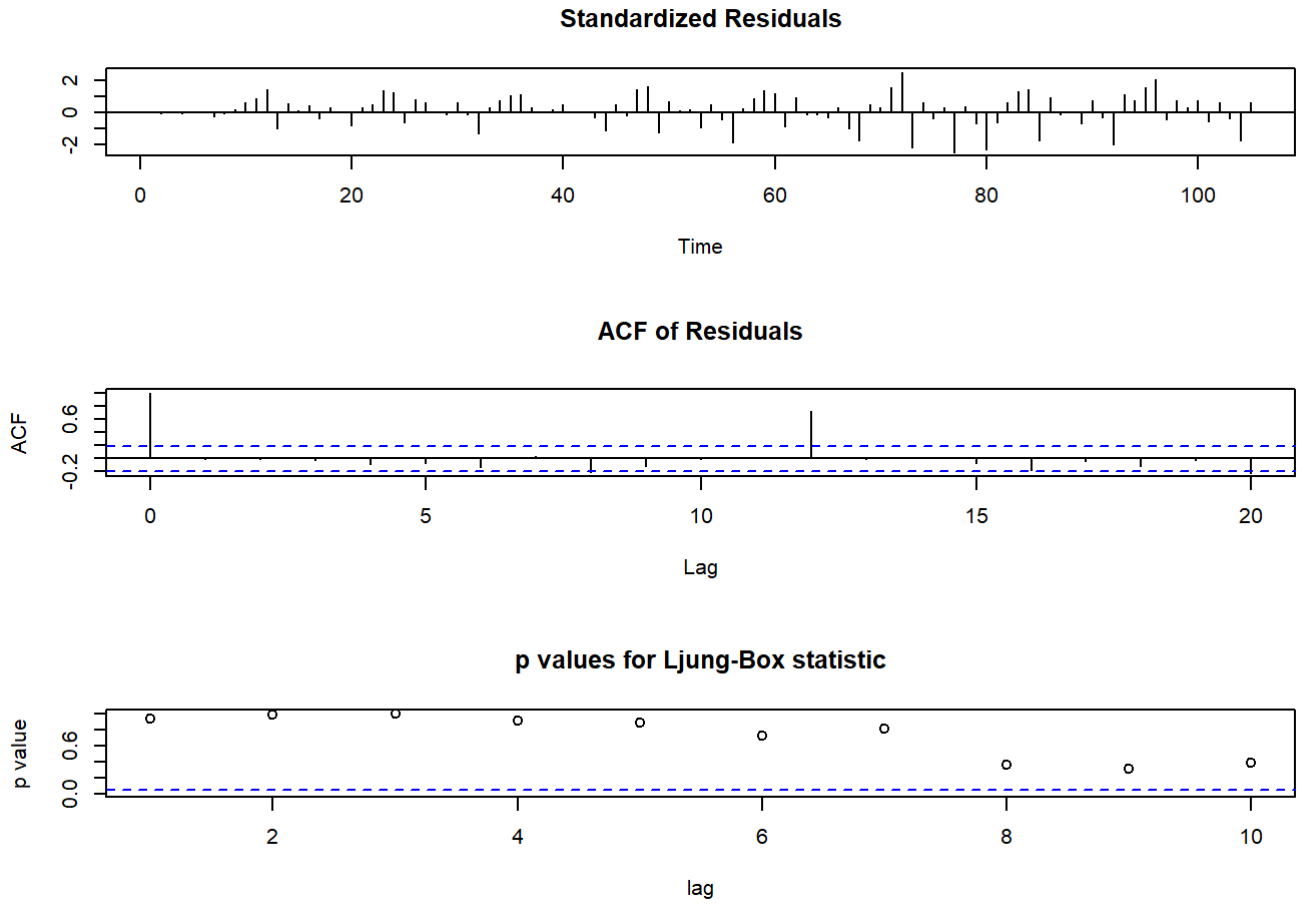
```
plot(density(residuals))
```




```
# Perform a Ljung-Box test on the residuals
Box.test(residuals, lag = 10, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 10.583, df = 10, p-value = 0.3909
```

```
#LBQPlot(residuals, lag.max = 20, SquaredQ = FALSE)
tsdiag(arima_model)
```

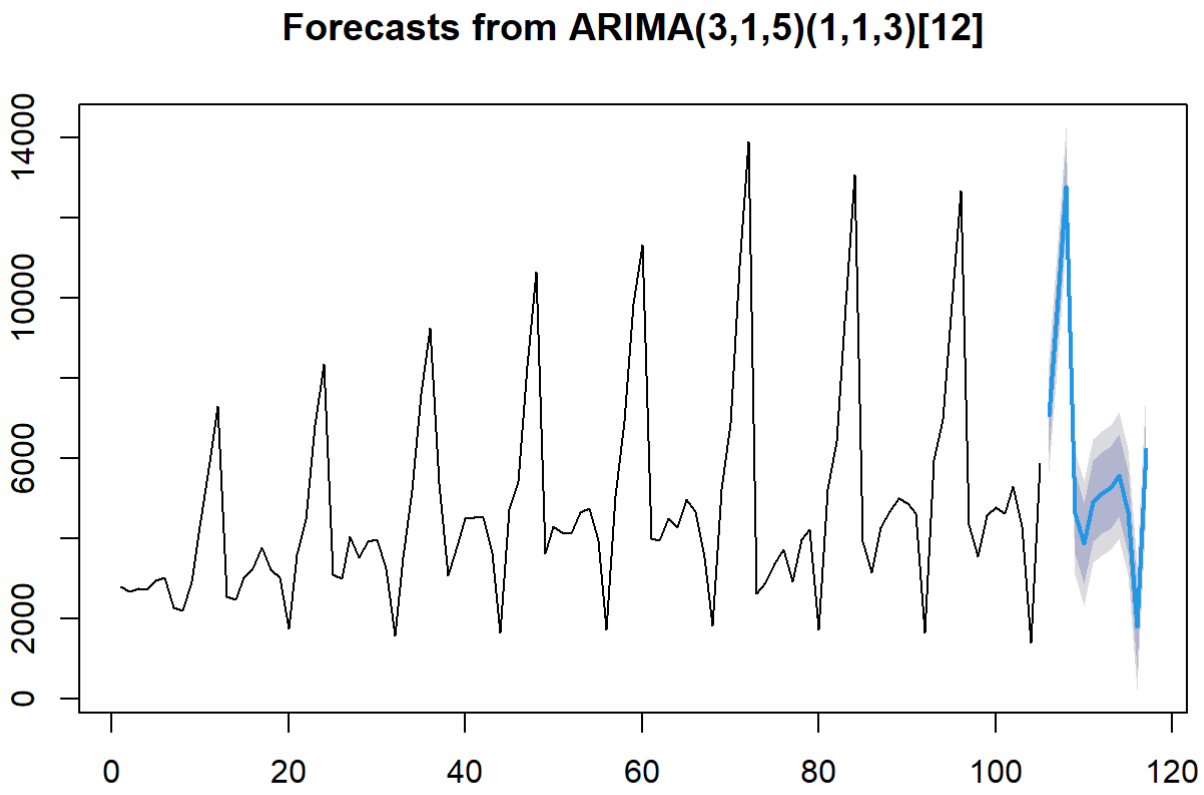


the code performs a Ljung-Box test on the residuals using the “Box.test” function to formally test for residual autocorrelation. The test statistic is compared to a chi-squared distribution with degrees of freedom equal to the number of lags specified in the test. A significant p-value (i.e., less than 0.05) indicates evidence of residual autocorrelation, which suggests that the model may be misspecified and may require further modification.

In this case, I got a p-value more than 0.05 in the Ljung-Box test. It suggests that this can be a good model.

Forecast Using the best model

```
# Load the "forecast" package
library(forecast)

# Fit a SARIMA(3,1,3)(1,1,3) model to the time series
sarima_model <- Arima(data_clean$sales, order = c(3,1,5), seasonal = list(order = c(1,1,3), p
eriod = 12))

# Generate a 12-month forecast from the SARIMA model
forecast_data <- forecast(sarima_model, h =12)

# Plot the forecasted values
plot(forecast_data)
```



Forecast looks preety much good and now I am moving on non-seasonal dataset.