Appendix

1. Data prepare

```
# Load packages
library(tsdl)
tsdl

## Time Series Data Library: 648 time series
##
##                           Frequency
## Subject                    0.1 0.25   1  4  5  6  12 13 52 365 Total
##   Agriculture                0    0  37  0  0  0   3  0  0   0    40
##   Chemistry                  0    0   8  0  0  0   0  0  0   0     8
##   Computing                  0    0   6  0  0  0   0  0  0   0     6
##   Crime                      0    0   1  0  0  0   2  1  0   0     4
##   Demography                 1    0   9  2  0  0   3  0  0   2    17
##   Ecology                    0    0  23  0  0  0   0  0  0   0    23
##   Finance                    0    0  23  5  0  0  20  0  2   1    51
##   Health                     0    0   8  0  0  0   6  0  1   0    15
##   Hydrology                  0    0  42  0  0  0  78  1  0   6   127
##   Industry                   0    0   9  0  0  0   2  0  1   0    12
##   Labour market              0    0   3  4  0  0  17  0  0   0    24
##   Macroeconomic              0    0  18 33  0  0   5  0  0   0    56
##   Meteorology                0    0  18  0  0  0  17  0  0  12    47
##   Microeconomic              0    0  27  1  0  0   7  0  1   0    36
##   Miscellaneous              0    0   4  0  1  1   3  0  1   0    10
##   Physics                    0    0  12  0  0  0   4  0  0   0    16
##   Production                 0    0   4 14  0  0  28  1  1   0    48
##   Sales                      0    0  10  3  0  0  24  0  9   0    46
##   Sport                      0    1   1  0  0  0   0  0  0   0     2
##   Transport and tourism      0    0   1  1  0  0  12  0  0   0    14
##   Tree-rings                 0    0  34  0  0  0   1  0  0   0    35
##   Utilities                  0    0   2  1  0  0   8  0  0   0    11
##   Total                      1    1 300 64  1  1 240  3 16  21   648

# Choosing data
data <- subset(tsdl,12,"Sales")
data

## Time Series Data Library: 24 Sales time series with frequency 12
##
##          Frequency
## Subject 12
##   Sales 24

sales_ts <- data[[2]]
sales_ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1965  38  44  53  49  54  57  51  58  48  44  42  37
## 1966  42  43  53  49  49  40  40  36  29  31  26  23
## 1967  29  32  41  44  49  47  46  47  43  45  34  31
## 1968  35  43  46  46  43  41  44  47  41  40  32  32
## 1969  34  40  43  42  43  44  39  40  33  32  31  28
## 1970  34  29  36  42  43  44  44  48  45  44  40  37
## 1971  45  49  62  62  58  59  64  62  50  52  50  44
## 1972  51  56  60  65  64  63  63  72  61  65  51  47
## 1973  54  58  66  63  64  60  53  52  44  40  36  28
## 1974  36  42  53  53  55  48  47  43  39  33  30  23
## 1975  29  33  44  54  56  51  51  53  45  45  44  38
```

```
# View data
ts("sales_ts")
```

```
## Time Series:
## Start = 1
## End = 1
## Frequency = 1
## [1] sales_ts
```

```
tsp(sales_ts)
```

```
## [1] 1965.000 1975.917   12.000
```

```
str(sales_ts)
```

```
##  Time-Series [1:132] from 1965 to 1976: 38 44 53 49 54 57 51 58 48 44 ...
##  - attr(*, "source")= chr "Abraham & Ledolter (1983)"
##  - attr(*, "description")= chr "Monthly sales of U.S. houses (thousands) 1
965 - 1975"
##  - attr(*, "subject")= chr "Sales"
```

```
summary(sales_ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   23.00   39.00   44.00   45.36   52.25   72.00
```

```
# Check any missing value
summary(is.na(sales_ts))
```

```
##    Mode   FALSE
## logical     132
```

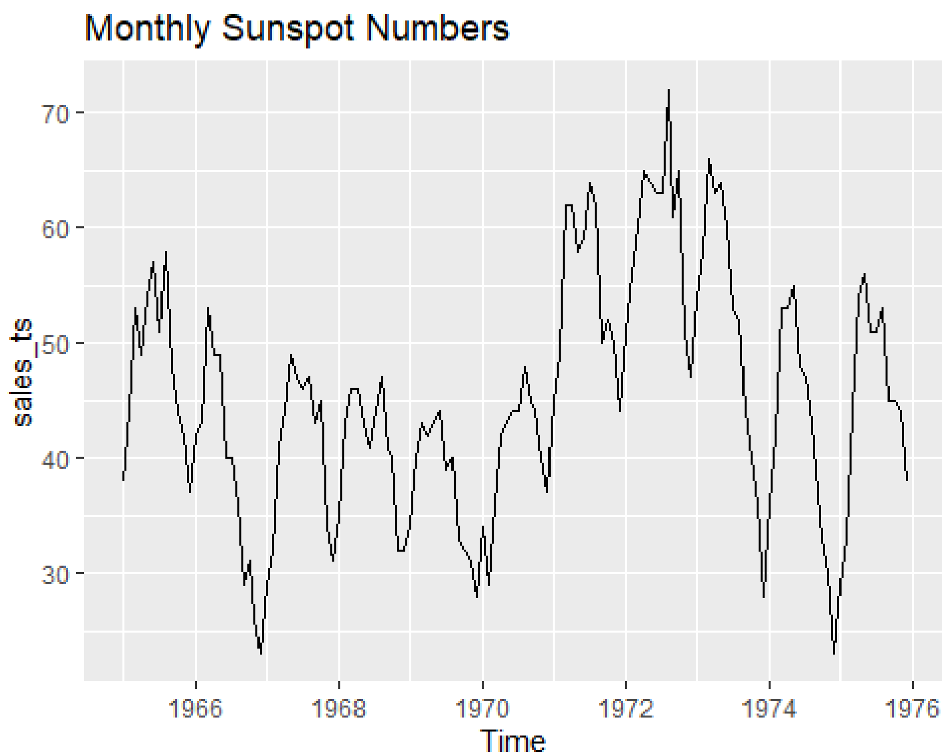
2. Exploraroty data analysis

```
library(forecast)
```

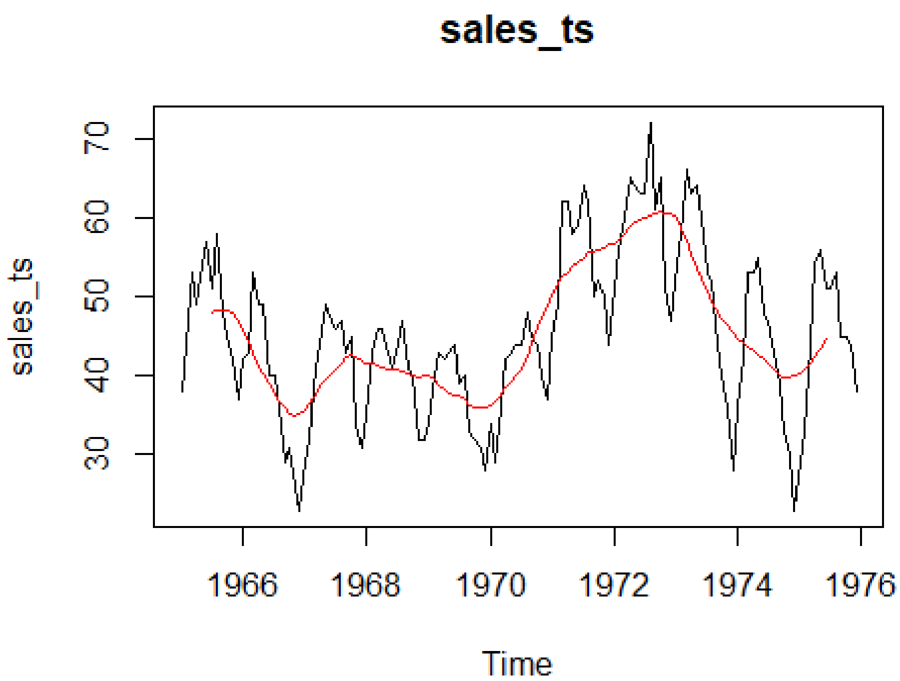
```
## Warning: 程辑包'forecast'是用 R 版本 4.2.3 来建造的
```

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

```
library(ggplot2)
# View data
autoplot(sales_ts , main="Monthly Sunspot Numbers")
```



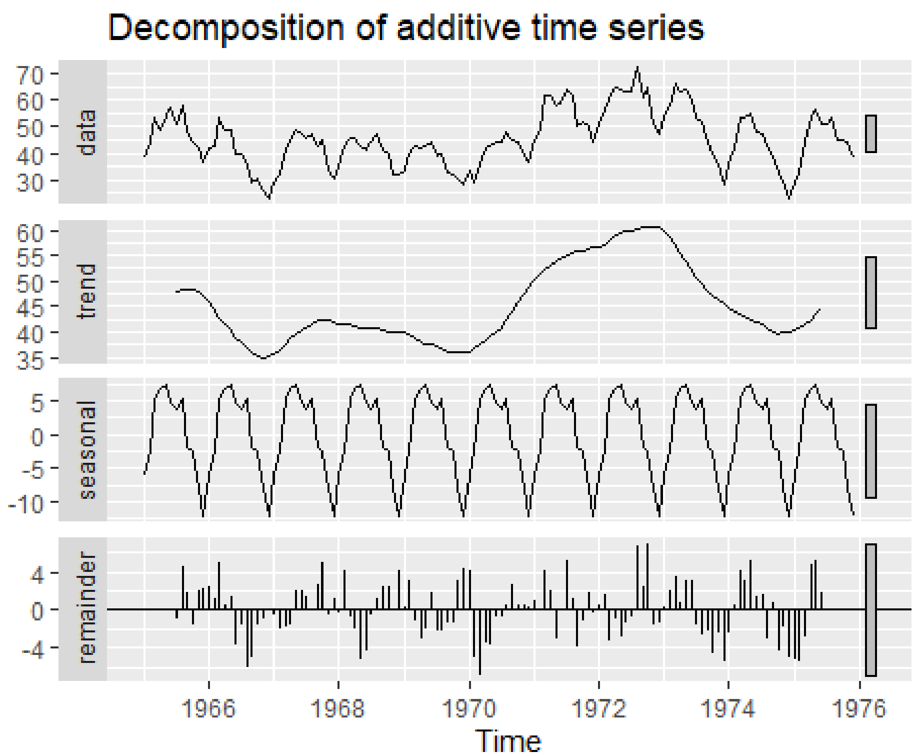

```
# Finding trend of a time series
ma_data <- ma(sales_ts, order = 12, centre = T)

# Plot the original data and the moving average
plot(sales_ts, main = "sales_ts")
lines(ma_data, col = "red")
```



3.Time series decomposition

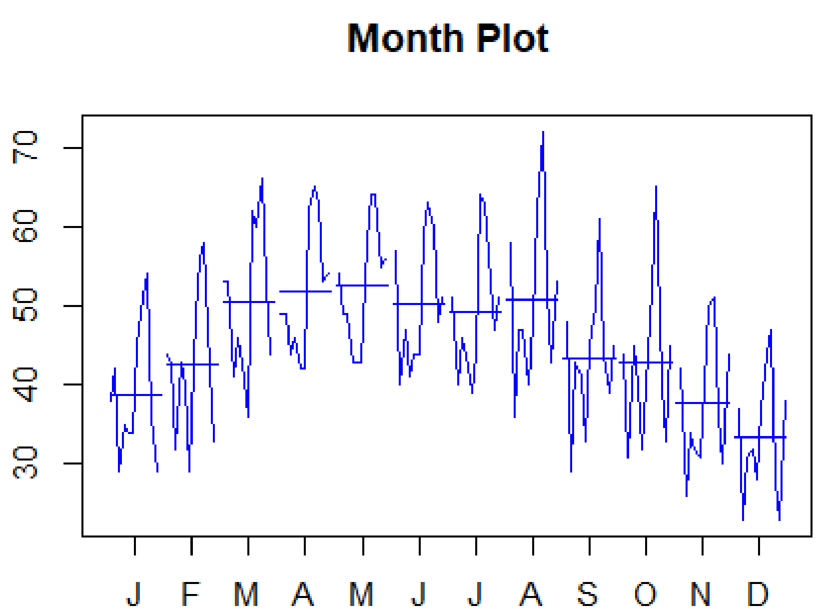
3.1 Classical Decomposition

```
# Perform classical decomposition
decomp_ts <- decompose(sales_ts)
# Plot the original and decomposed time series
autoplot(decomp_ts)
```

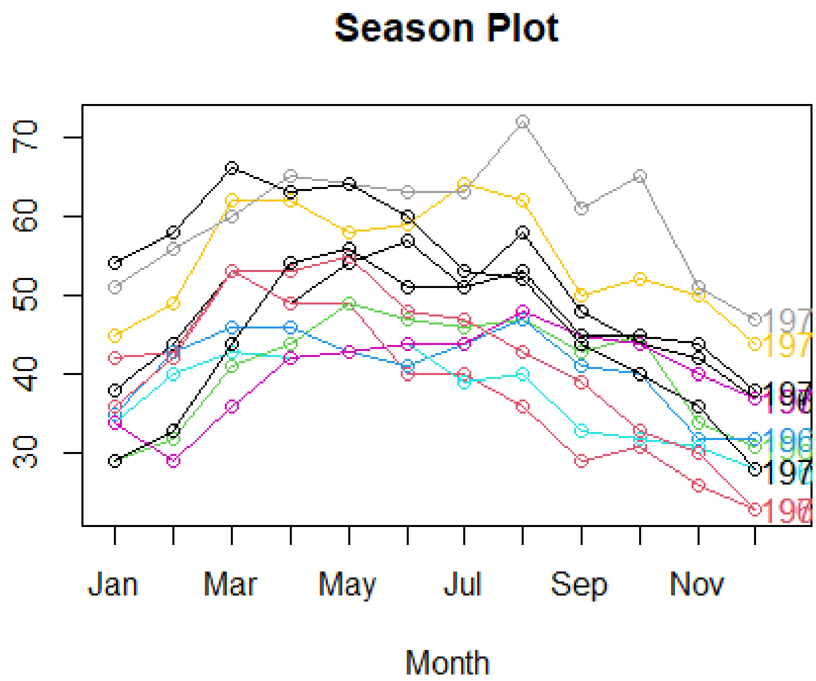


there are still repeated patterns that suggest there is a seasonality.

```
monthplot(sales_ts, xlab="", ylab="", main="Month Plot", col = "blue")
```

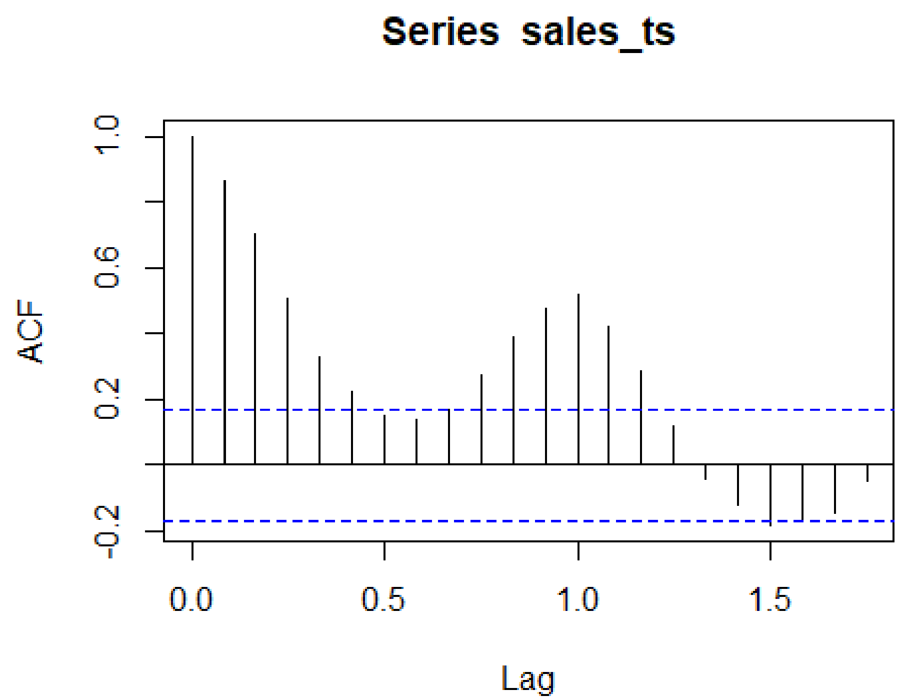


```r
seasonplot(sales_ts, year.labels="TRUE", main="Season Plot", col = 1:10)
```

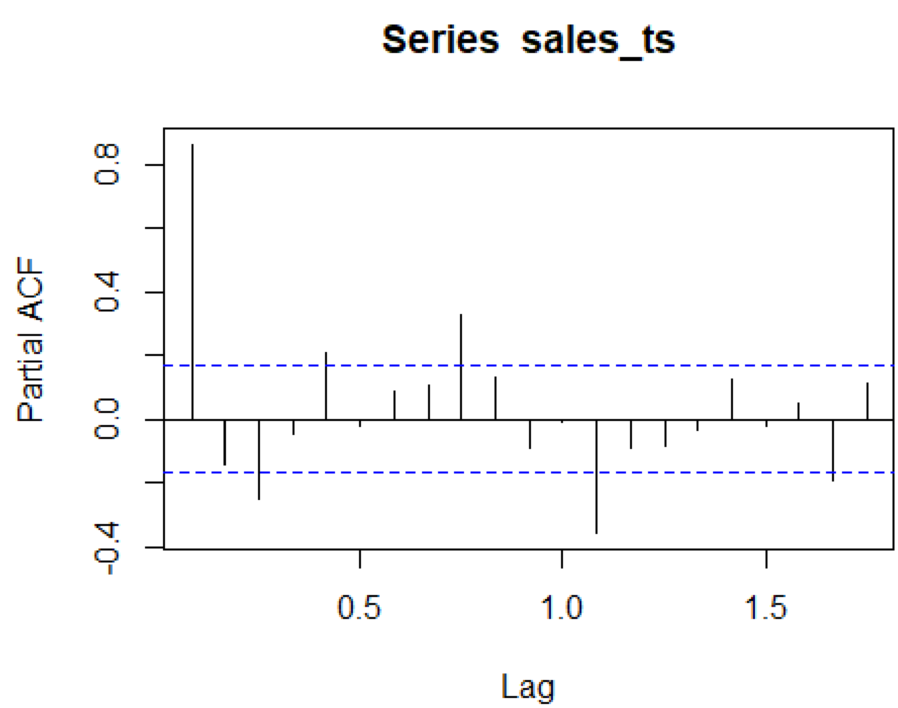


4. Model selection

```r
# Autocorrelation function (ACF) and partial autocorrelation function (PACF)
acf(sales_ts)
```




```
pacf(sales_ts)
```



```
# Test stationay
library(tseries)
kpss_test <- kpss.test(sales_ts)
print(kpss_test)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sales_ts
## KPSS Level = 0.41574, Truncation lag parameter = 4, p-value = 0.07037
```

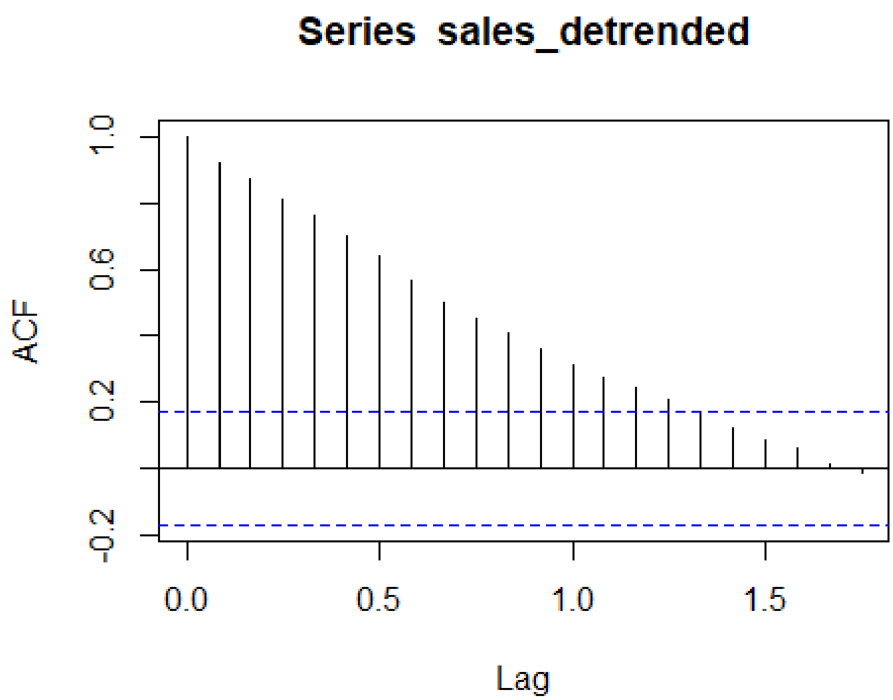
The dataset is non-stationary.

```
# Remove the seasonal component
sales_detrended <- sales_ts - decomp_ts$seasonal

# Test stationay
kpss_test <- kpss.test(sales_detrended)
print(kpss_test)
```

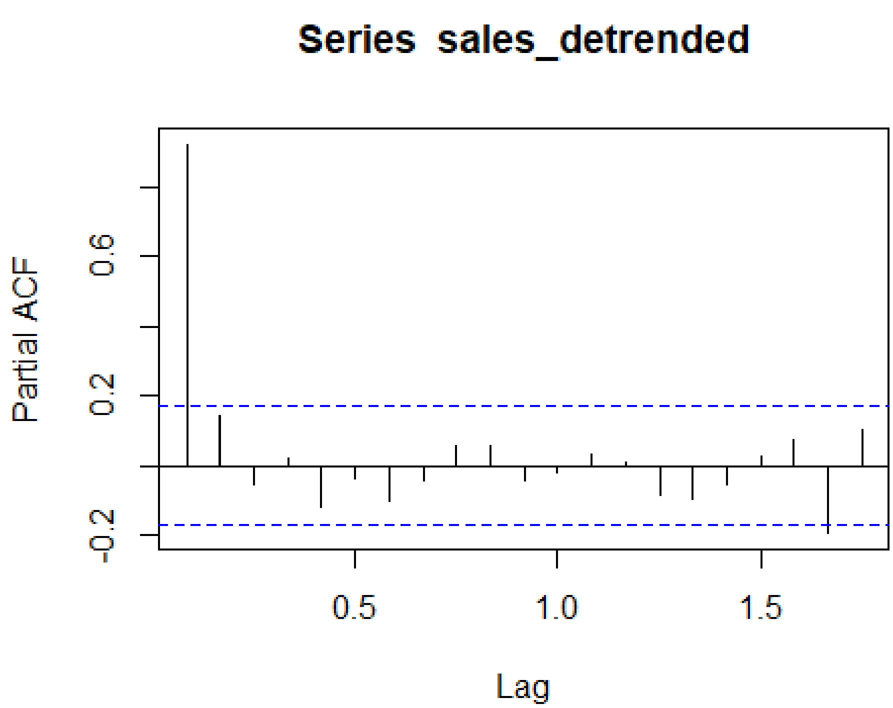
```
##
##  KPSS Test for Level Stationarity
##
## data:  sales_detrended
## KPSS Level = 0.57455, Truncation lag parameter = 4, p-value = 0.02495
```

Now it is stationary.

```
# Autocorrelation function (ACF) and partial autocorrelation function (PACF)
acf(sales_detrended)
```




```
pacf(sales_detrended)
```



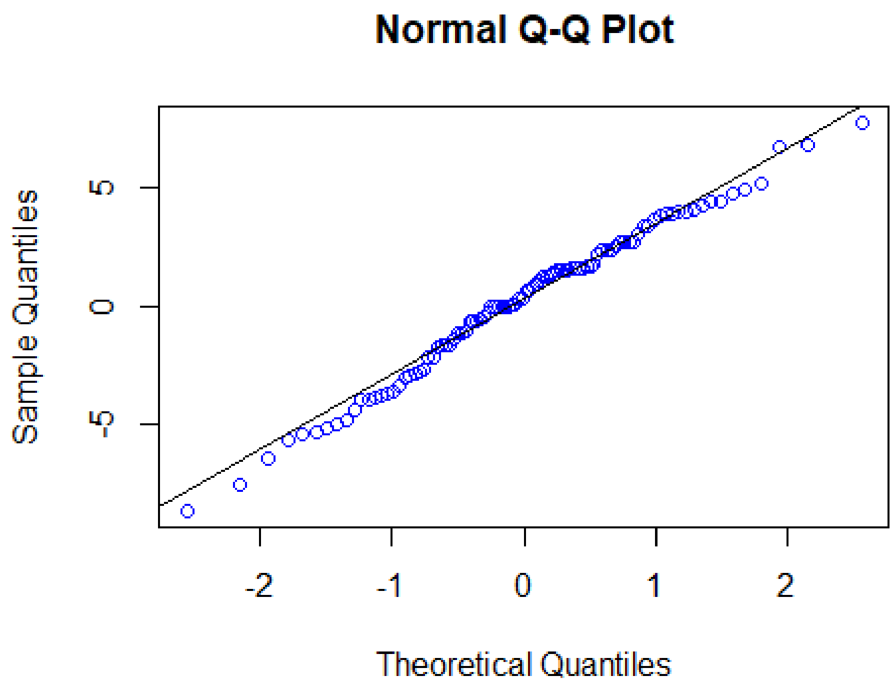
```r
# Split the data into training and testing sets
train_len <- floor(length(sales_detrended) * 0.7) # 70% for training
train <- window(sales_detrended, end = c(1972, 12))
test <- window(sales_detrended, start = c(1973, 1))
```

4.1 Model 1: ARIMA model

```r
# Fit an ARIMA model
library(forecast)
arima_model  <- arima(train, order = c(1,1,0))

# Print the model summary
summary(arima_model )
```

```
##
## Call:
## arima(x = train, order = c(1, 1, 0))
##
## Coefficients:
##           ar1
##       -0.2259
## s.e.   0.0996
##
## sigma^2 estimated as 10.83:  log likelihood = -247.97,  aic = 499.94
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.1899728 3.273093 2.636287 0.03996225 5.991369 0.9751986
##                     ACF1
## Training set -0.01402288
```

```r
# EVALUATING MODEL FIT
qqnorm(arima_model$residuals, col="blue")
qqline(arima_model$residuals)
```




```
Box.test(arima_model$residuals, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  arima_model$residuals
## X-squared = 0.019474, df = 1, p-value = 0.889
```

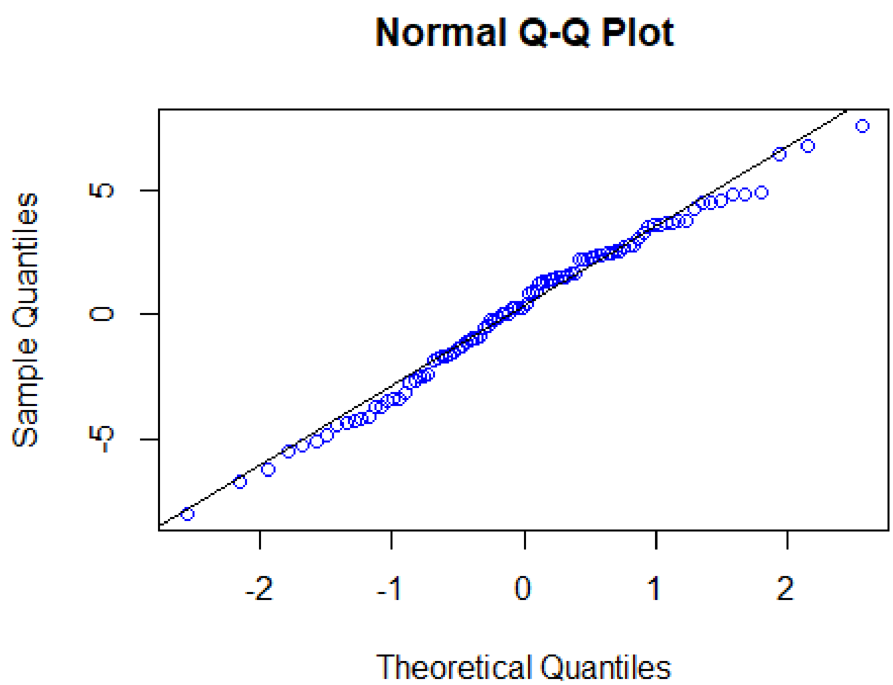
let the function automatically selects the best ARIMA or SARIMA model based on the AIC (Akaike Information Criterion) value.

4.2 Model 2: Seasonal ARIMA Modeling

```
# Fit a seasonal ARIMA model
sarima_model <- auto.arima(train, seasonal = TRUE)

# Print the model summary
summary(sarima_model)

## Series: train
## ARIMA(0,1,1)(0,0,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.2528  -0.1960
## s.e.   0.0970   0.1219
##
## sigma^2 = 10.69:  log likelihood = -246.61
```

```
## AIC=499.21   AICc=499.48   BIC=506.87
##
## Training set error measures:
##                    ME     RMSE     MAE        MPE     MAPE      MASE
## Training set 0.245236 3.218307 2.65721 0.09754921 6.025415 0.3504013
##                     ACF1
## Training set -0.01460636
```

```
# EVALUATING MODEL FIT
qqnorm(sarima_model$residuals, col="blue")
qqline(sarima_model$residuals)
```




```
Box.test(sarima_model$residuals, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  sarima_model$residuals
## X-squared = 0.021128, df = 1, p-value = 0.8844
```

4.3 Model 3: single exponential model

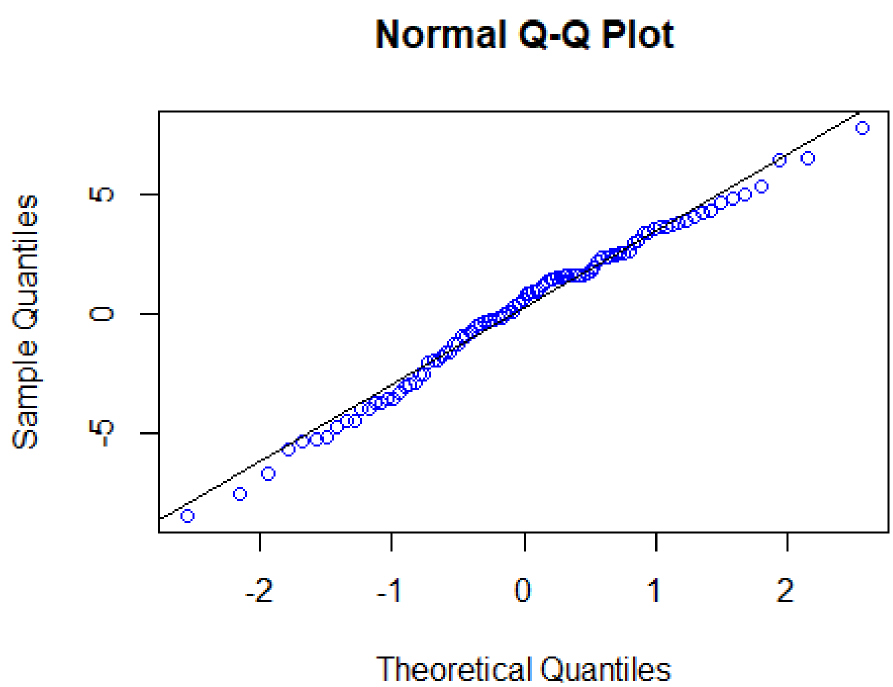
```
# Fit the SES model with seasonal pattern to the training data using the opti
mal values of the smoothing parameters
ses_model <- ses(train, alpha = sarima_model$model$alpha, initial = "simple",
 h = length(test), season = "additive")
ses_model$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = train, h = length(test), initial = "simple", alpha = sarima_model
$model$alpha,
##
##  Call:
##      season = "additive")
##
##   Smoothing parameters:
##     alpha = 0.7596
##
##   Initial states:
##     l = 44.2174
##
##   sigma:  3.2689
```

```
accuracy(ses_model)
```

```
##                         ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.2100648 3.268931 2.653659 0.06047718 6.040407 0.3499331
##                       ACF1
## Training set -0.003876693
```

```
# EVALUATING MODEL FIT
qqnorm(ses_model$residuals, col="blue")
qqline(ses_model$residuals)
```



```
Box.test(ses_model$residuals, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  ses_model$residuals
## X-squared = 0.0014883, df = 1, p-value = 0.9692
```

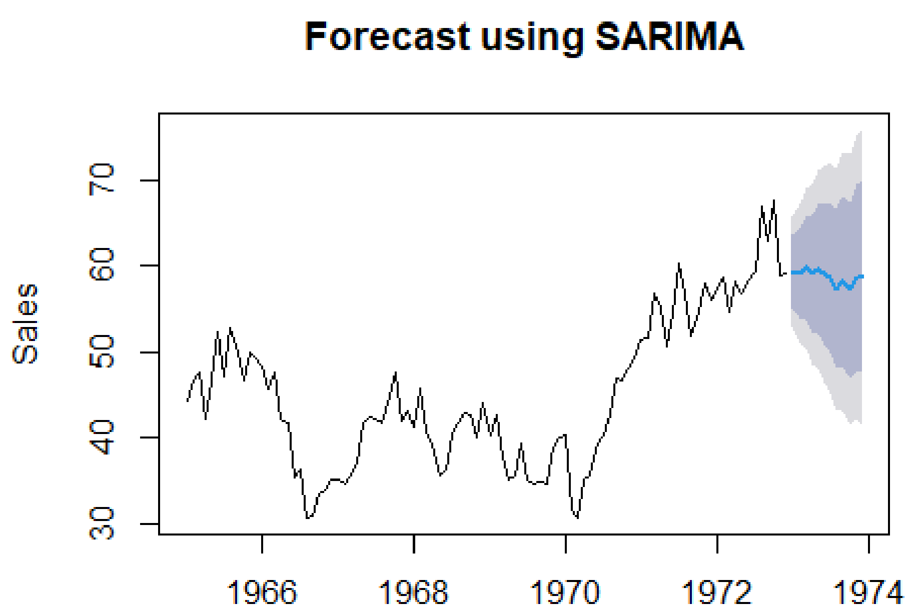
5. Model selectoin

```
# Use the fitted models to forecast on the test set
arima_forecast <- forecast(arima_model, h = length(test))
sarima_forecast <- forecast(sarima_model, h = length(test))
ses_forecast <- forecast(ses_model, h = length(test), season = "additive")

# Calculate the accuracy metrics for each model on the test set
accuracy(arima_forecast, test)

##                       ME      RMSE       MAE         MPE      MAPE      M
ASE
## Training set   0.1899728  3.273093  2.636287   0.03996225  5.991369 0.3476
422
## Test set     -12.9139646 14.688901 13.144944 -30.88706887 31.268733 1.7333
992
##                     ACF1 Theil's U
## Training set -0.01402288        NA
## Test set      0.84515483  5.167752

accuracy(sarima_forecast, test)

##                      ME      RMSE      MAE         MPE      MAPE      MAS
E
## Training set   0.245236  3.218307  2.65721   0.09754921  6.025415 0.350401
3
## Test set     -12.419870 14.156561 12.60856 -29.73419974 30.046009 1.662667
1
##                     ACF1 Theil's U
## Training set -0.01460636        NA
## Test set      0.84758533  4.996265

accuracy(ses_forecast, test)

##                       ME      RMSE       MAE         MPE      MAPE      M
ASE
## Training set   0.2100648  3.268931  2.653659   0.06047718  6.040407 0.3499
331
## Test set     -13.3408582 15.065423 13.500368 -31.83226719 32.095780 1.7802
683
##                      ACF1 Theil's U
## Training set -0.003876693        NA
## Test set      0.845197829  5.292789
```

The choice of the best model depends on the specific forecasting problem and the measure of forecast accuracy that is most important for the decision maker.

6. Forecasting

```
# Make a forecast for the next year
forecast <- forecast(sarima_model, h = 12)

# Print the forecasted values
print(forecast$mean)

##           Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug
## 1973 59.31269 59.08797 59.68002 59.04813 59.48397 58.97190 58.58310 57.224
73
##           Sep      Oct      Nov      Dec
## 1973 58.14206 57.12353 58.65577 58.67770

# Plot the forecast and actual values
# Plot the forecast
plot(forecast, ylab="Sales", main="Forecast using SARIMA")
```




```
# Fit a seasonal ARIMA model
sarima_model1 <- auto.arima(sales_ts, seasonal = TRUE)

# Make a forecast for the next year
```

```
forecast <- forecast(sarima_model1, h = 12)

# Print the forecasted values
print(forecast$mean)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug
## 1976 44.47570 48.69200 58.33548 60.97031 62.37838 57.06210 54.05804 53.355
03
##           Sep      Oct      Nov      Dec
## 1976 46.12028 43.07137 40.34167 33.22366
```

```
# Plot the forecast and actual values
# Plot the forecast
autoplot(forecast, ylab="Sales", main="Forecast using SARIMA")
```