Time Series Analysis of Gasoline demand in Ontario from 1960-1975

Jenny Zhang
Pstat 174
June 13, 2019

Abstract

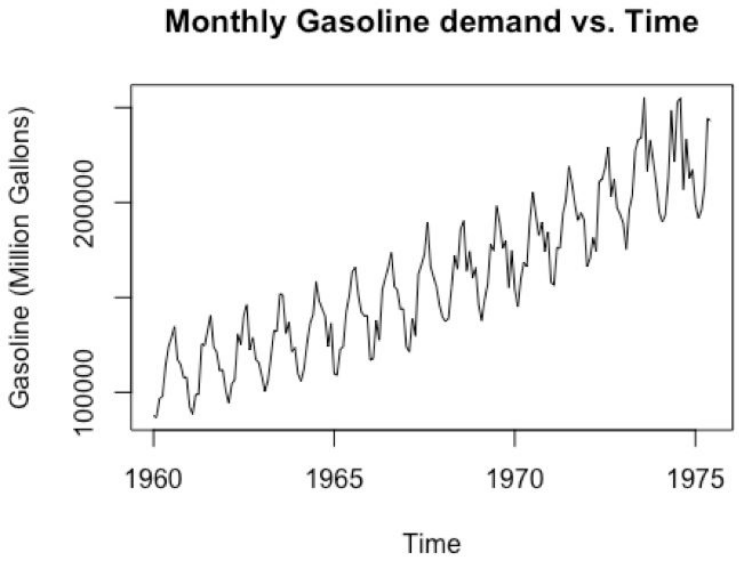
In this time series analysis, I will analyze the data collected on monthly gasoline demand in Ontario from 1960 to 1975. I transformed and differenced the data to predict a working model that would help forecast future gasoline demand by the millions of gallons. Ultimately, a Seasonal ARIMA model was used to help fit the data and predict future observations.

Introduction

The main purpose of this analysis was to determine a working model that best fits the dataset to help predict and forecast future observations. The dataset, "Monthly gasoline demand Ontario gallon millions 1960 - 1975" by Abraham & Ledolter (1983) is collected from Time Series Data Library by Yuanbo Wang. The data depicts monthly observations of gasoline demand by the millions of gallons. In order to test the forecasting abilities of the model, I removed the last 6 observations from the dataset and plotted the remaining dataset in RStudio. I then analyzed the data using the R programming language. Understanding that there is a trend and seasonality in the data, some modifications had to be made in order to find the final model. Firstly, a log transformation was used to decrease the variance. Then, differencing was done at lag = 1 and lag = 12 to remove trend and seasonality respectively. After, the ACF and PACF were analyzed to help determine potential parameters for the model. Several models were then tested to find the best fit model through Akaike information criterion and number of parameters. Then, we boiled down the models to two fits and performed diagnostic testing to determine the best fitting model. The final model was then used to help forecast the last 6 observations of gasoline demand for the months of July-December that was initially removed. Overall, most steps of the process were performed smoothly. One issue that occurred was that my final model failed the McLeod-Li test for the square of the model's residuals. However, this was the only test the model failed, and my final model still fared well compared to other models. Additionally, another problem that occurred was that a couple of the observed points from the original data did not fit in the confidence interval proposed by our final model. This was able to be overlooked because one point was almost on the line of the confidence interval, and most of the observed data was in fact between the upper and lower bounds of the confidence interval making my model fairly accurate.

Analysis

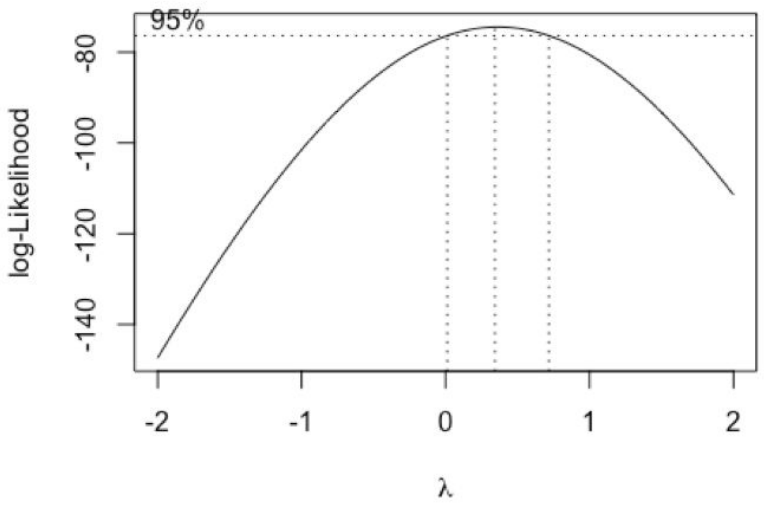
I. Initial Data Observations



When observing the original data, I can see that the data has an upwards growing trend, along with 15 seasonal spikes for each of the 15 years from 1960-1975. We know from this analysis that we will need to transform, detrend and deseasonalize the data to obtain a stationary time series.

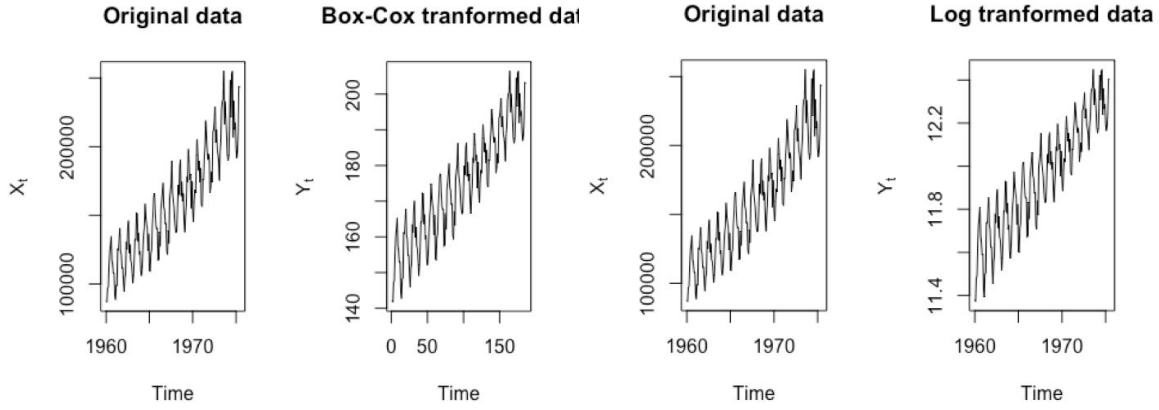
II. Transformations

To begin the process of making the data stationary, I wanted to transform the data to minimize the variance. First, I used the boxcox function to determine λ based on log-likelihood.



From the plot, I can see that the 95% confidence interval for log-likelihood includes the λ value of 0, so I am leaning towards the direction of using $\lambda = 0$. In other words, I am leaning towards using a pure log transformation on the dataset. However, I wanted to try other transformations as well to ensure that a log transformation is my best outcome, such as the box-cox transformation that would utilize the following formula:

$$y_i^{(\lambda)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$



As I observe both the Box-Cox transformed data and the Log transformed data, I can already see that the regression line for the mean is straighter with the log transformation than with the

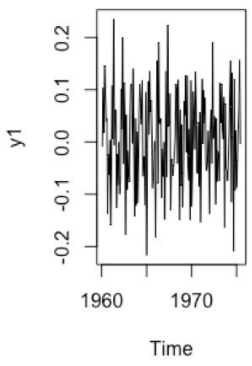
box-cox transformation. To confirm, I observed the variance of both data transformations. In addition, I added in the variance for a square root transformation for extra comparison:

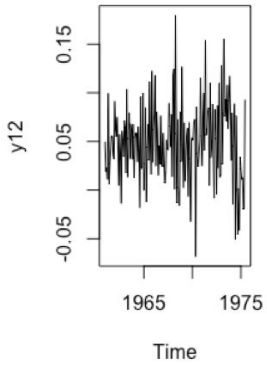
```
var(sale.ts)

## [1] 1600021030

var(sale.bc)

## [1] 236.3724Introductory Time Series Analysis with R by P. S. P.
Cowpertwait and A.V. Metcalfe, Springer

var(sale.log)

## [1] 0.06465378

var(sale.sqrt)

## [1] 2501.043
```

The variance for the original data was 1600021030 while the variance for the box-cox transformation, log transformation, and square root transformation are 236.3724, 0.06465378, and 2501.043 respectively. The smaller the variance, the better, so I decided to continue with the log transformed data.

III. Differencing

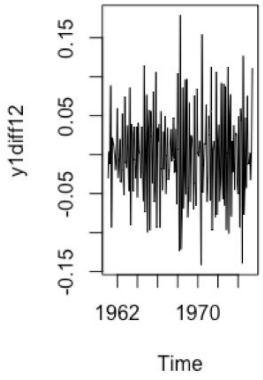
Now that the variance is more stabilized, as it jumped from 1600021030 to 0.06465378, I want to now continue with differencing to eliminate trend and seasonality. I differenced at both lag = 1 and lag = 12 to determine which way I would want to difference first.





The plot for the log transformed dataset ∇X_t that is differenced at lag = 1(y1) has a much more stationary and "white noise"-like pattern than the dataset $\nabla_{12} X_t$ differenced at lag = 12(y12). Therefore, I decided to difference at lag = 1 first. After differencing at lag = 1 and detrending

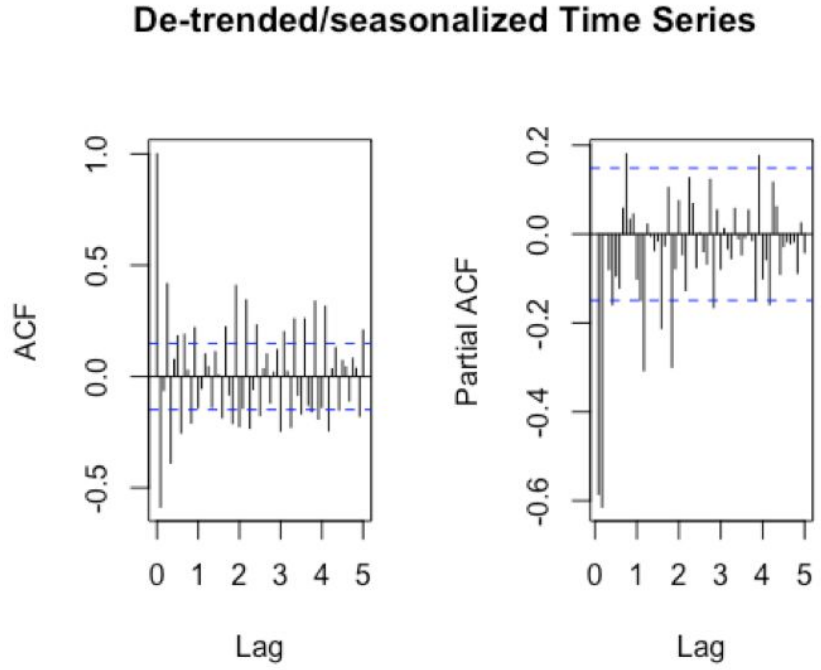
the series, my variance went down from 0.06465378 to 0.008146013. I now want to deseasonalize the series by differencing the data ∇X_t again at lag = 12.



The variance once again decreased from 0.008146013 for ∇X_t to 0.003695186 for $\nabla_{12} \nabla X_t$. I am now done with differencing, as I have already removed trend and seasonality. Any more differencing would lead to increased variance, and hence overdifferencing. The series is now stationary and can be analyzed for parameters.

IV. Preliminary model deducing

Now that I have a stationary series, I can decipher potentially what kind of model I want to use and its parameters. First I want to look at the ACF and PACF plots of my log transformed and differenced model:



From my observation, I can see that the ACF cuts off after lag = 1 and possibly lag = 2 for every 12 ticks, meaning there is a seasonal MA(1) or seasonal MA(2) counterpart. On the PACF plot, I can see the PACF plot tailing off. This leads me to believe that the model will have a (0,1,1) or (0,1,2) seasonal component. For the non seasonal part, I can see that PACF cuts off after lag = 2 and lag = 5, meaning that there is possibly an AR(2) or AR(5) counterpart.

V. Model testing and comparisons

From my observations in the deduction part, I have set up 16 potential models. I know that they will be SARIMA processes as they have both trend and seasonality. I've included the Akaike information criterion (AIC) for comparison, as the lower the AIC, the better fit a model is.

Number	Model	AIC
Model 1	$SARIMA(2,1,0)x(0,1,1)_{12}$	19.00323
Model 2	$SARIMA(2,1,1)x(0,1,1)_{12}$	19.00961
Model 3	$SARIMA(2,1,2)x(0,1,1)_{12}$	19.00937
Model 4	$SARIMA(2,1,3)x(0,1,1)_{12}$	18.98406
Model 5	$SARIMA(5,1,0)x(0,1,1)_{12}$	18.99541
Model 6	$SARIMA(5,1,1)x(0,1,1)_{12}$	18.99867
Model 7	$SARIMA(5,1,2)x(0,1,1)_{12}$	18.98334
Model 8	$SARIMA(5,1,3)x(0,1,1)_{12}$	18.95573
Model 9	$SARIMA(2,1,0)x(0,1,2)_{12}$	19.00047
Model 10	$SARIMA(2,1,1)x(0,1,2)_{12}$	19.00957
Model 11	$SARIMA(2,1,2)x(0,1,2)_{12}$	19.00802
Model 12	$SARIMA(2,1,3)x(0,1,2)_{12}$	18.98626
Model 13	$SARIMA(5,1,0)x(0,1,2)_{12}$	18.99128
Model 14	$SARIMA(5,1,1)x(0,1,2)_{12}$	18.99443
Model 15	$SARIMA(5,1,2)x(0,1,2)_{12}$	18.9663
Model 16	$SARIMA(5,1,3)x(0,1,2)_{12}$	18.95221

The models I observed with the lowest AIC are 18.95 for Model 16, 18.97 for Model 15, 18.896 for Model 12, 18.956 for Model 8, 18.983 for Model 7 and 18.984 for Model 4. By the law of parsimony, we want to choose the model with the fewest parameters, so we decide on Model 4 with 6 parameters. I will continue with diagnostics for Model 4, and will include the diagnostics for Model 16 for comparison, as it had the lowest AIC score.

VI. Model Diagnostics

To confirm that my model is a working and satisfactory model, I will run diagnostic tests. Namely, I will use the Box-Pierce, Box-Lyung and McLeod-Li tests to determine similarity to white noise and I will use the Shapiro-Wilk test, the histogram plot and the quantile-quantile plot to determine normality.

```
Box.test(model4$residuals, lag = 14, type = c("Box-Pierce"), fitdf = 6)
```

```
##
##  Box-Pierce test
##
## data:  model4$residuals
## X-squared = 10.051, df = 8, p-value = 0.2614
```

```
Box.test(model4$residuals, lag =14, type = c("Ljung-Box"), fitdf = 6)
```

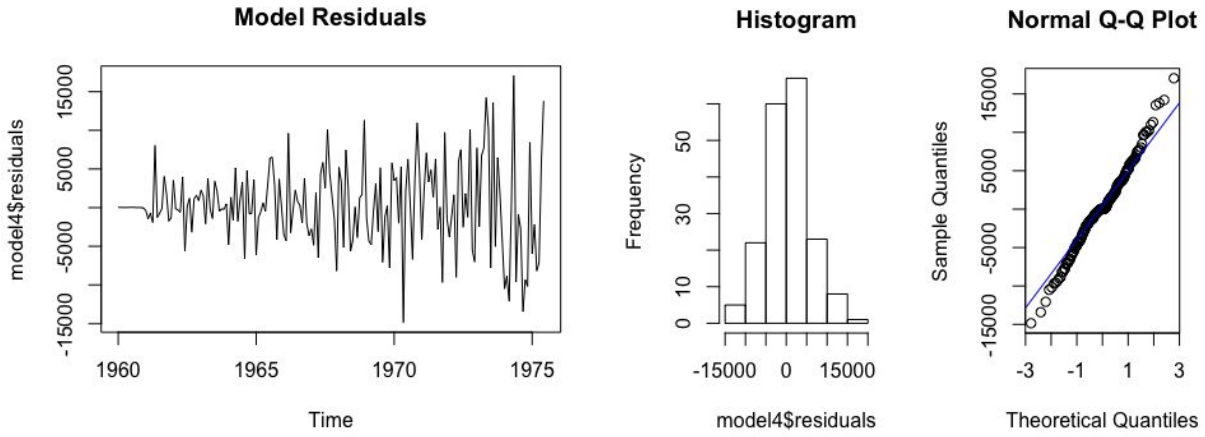
```
##
##  Box-Ljung test
##
## data:  model4$residuals
## X-squared = 10.639, df = 8, p-value = 0.223
```

```
Box.test((model4$residuals)^2, lag =14, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  (model4$residuals)^2
## X-squared = 142.16, df = 14, p-value < 2.2e-16
```

```
shapiro.test(model4$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4$residuals
## W = 0.9889, p-value = 0.1559
```



For Model 4, the residuals passed the tests for Box-Pierce, Box-Lyung, and Shapiro-Wilk with p-values all above the critical value of 0.05. When observing the plot of the residuals, histogram and Q-Q plot, the residuals seem to largely follow white noise and normality. The only test that Model 4 failed is the McLeod-Li testing the square of the residuals. However, the model passes all the other diagnostic tests leading me to believe this is a fit model.

To compare, I've included diagnostics for Model 16:

```
Box.test(model16$residuals, lag = 14, type = c("Box-Pierce"), fitdf = 10)
```

```
##
##  Box-Pierce test
##
## data:  model16$residuals
## X-squared = 12.013, df = 4, p-value = 0.01725
```

```
Box.test(model16$residuals, lag =14, type = c("Ljung-Box"), fitdf = 10)
```

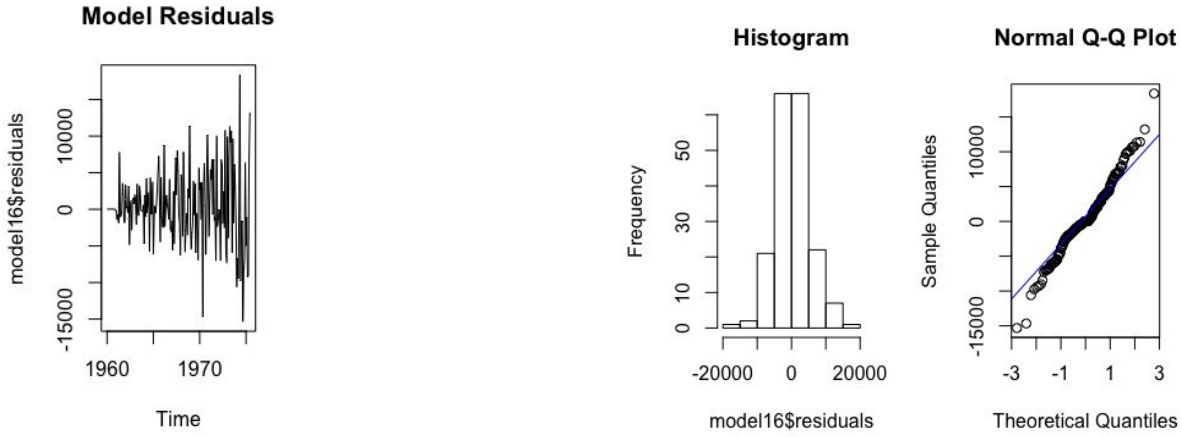
```
##
##  Box-Ljung test
##
## data:  model16$residuals
## X-squared = 12.72, df = 4, p-value = 0.01273
```

```
Box.test((model16$residuals)^2, lag =14, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  (model16$residuals)^2
## X-squared = 114.05, df = 14, p-value < 2.2e-16
```

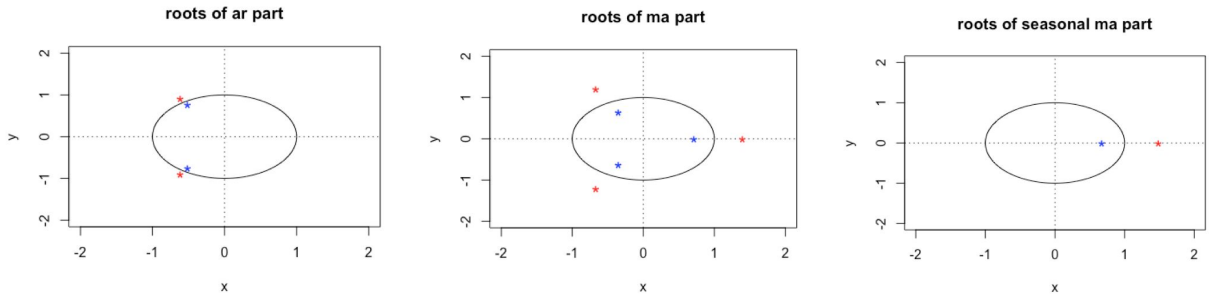
```
shapiro.test(model16$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model16$residuals
## W = 0.98061, p-value = 0.01099
```



For Model 16, the residuals failed the tests for Box-Lyung, Box-Pierce, McLeod-Li and Shapiro-Wilk with p-values below the critical value of 0.05 for all tests. Despite having the lowest AIC, Model 16 failed to compare with Model 4 in terms of white noise likeliness and normality. This can be attributed to the abundance of parameters in the model.

Therefore, I concluded that Model 4 is the best fit for the dataset. To confirm, I wanted to make sure that Model 4 is causal and invertible using the roots test:



All of my roots are outside of the unit circle, so the model SARIMA(2,1,3)x$(0,1,1)_{12}$ is indeed causal and invertible.

Therefore, after taking all my coefficients into account:

```
## Coefficients:
##           ar1      ar2      ma1     ma2      ma3     sma1
##       -1.0298  -0.8367  -0.0118  0.0217  -0.3765  -0.6736
## s.e.   0.0761   0.0864   0.1290  0.0979   0.1427   0.0797
```

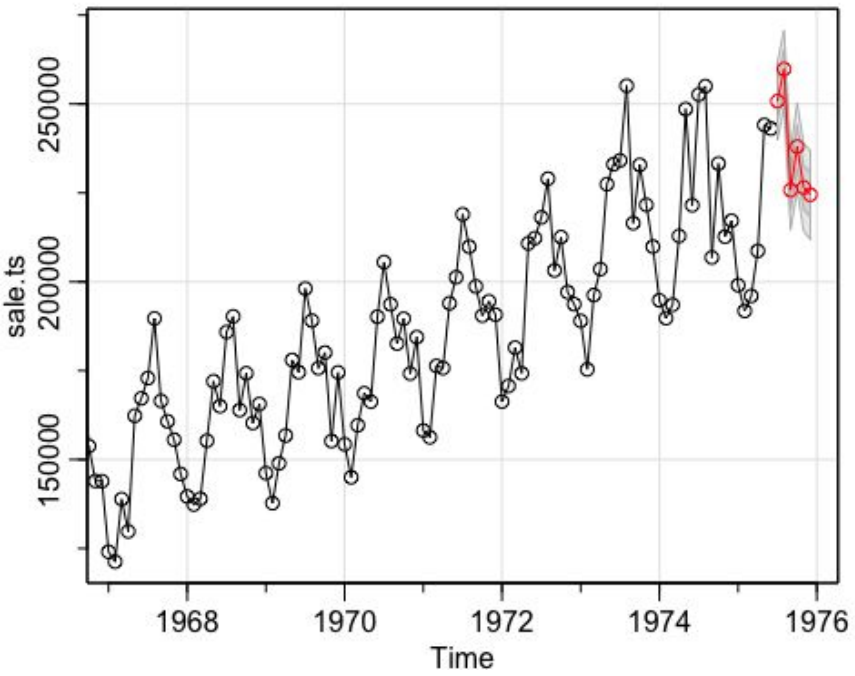
```
## sigma^2 estimated as 30151984:  log likelihood = -1739.53,  aic = 3493.07
```

My final model would be:

$$(1+1.0298_{(0.0761)}B+0.8367_{(0.0864)}B^2)\nabla_{12}\nabla X_t$$
$$=(1-0.0118_{0.129}B+0.0217_{0.0979}B^2-0.3765_{0.1427}B)(1-0.6736_{0.0797}B^{12})Z_t$$
$$\widehat{\sigma} = 30151984$$

VII. Forecasting

As I have removed the last 6 observations from the original dataset, I will now use my final model to predict the next 6 observations of gasoline demand in 1975 from July to December. Using the sarima forecast function, I obtained the following plot:



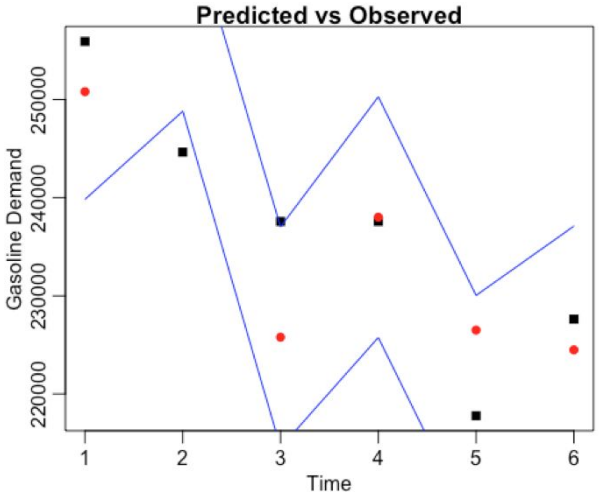
My predictions alongside their standard errors are as follows:

```
## $pred
##           Jul      Aug      Sep      Oct      Nov      Dec
## 1975 250804.0 259798.2 225782.5 238013.9 226507.4 224485.9
##
## $se
##           Jul      Aug      Sep      Oct      Nov      Dec
## 1975 5491.092 5495.828 5622.363 6132.156 6132.255 6305.060
```

To obtain the confidence intervals, I multiplied the standard errors by 2 and added and subtracted from the predictions. To show the comparison of the observed points with the predicted points, I've created a table:

Month	July	August	September	October	November	December
Predicted Values	250804.0	259798.2	225782.5	238013.9	226507.4	224485.9
Confidence Intervals	(239821.8, 261786.24)	(248806.5, 270789.85)	(214537.8, 237027.22)	(225749.6, 250278.21)	(205510.5, 230039.51)	(211875.8, 237096.02)
Observed Values	255918	244642	237579	237579	217775	227621

For the most part, my observed values were within the confidence intervals. August is a little bit further down from the lower confidence interval while September is practically almost on the line, though just a little bit higher than the upper confidence interval.
To visualize this, I plotted the following graph with the black points representing observed values, red points representing predicted values, and blue lines representing upper and lower bounds of the confidence interval:



For the most part, my model is very accurate and does a relatively precise job of forecasting future observations of gasoline demand per month.

Conclusion

In conclusion, the model I obtained was a SARIMA(2,1,3)x$(0,1,1)_{12}$ model with the form:

$$(1+1.0298_{(0.0761)}B+0.8367_{(0.0864)}B^2)\nabla_{12}\nabla X_t=(1-0.0118_{0.129}B+0.0217_{0.0979}B^2-0.3765_{0.1427}B)(1-0.6736_{0.0797}B^{12})Z_t$$

This model was then used to help predict the gasoline demand in Ontario from July 1975 to December 1975. Ultimately, the model is fairly useful and accurate though two observed values were not as close to our predicted values compared to the other points.

References

Dataset collected tsdl: Time Series Data Library by Yuanbo Wang

Introduction to Time Series and Forecasting, by P. Brockwell and R. Davis, Springer

Time Series Analysis with R Examples, by R. H. Shumway and D. S. Stoffer, Springer.

Introductory Time Series Analysis with R, by P. S. P. Cowpertwait and A.V. Metcalfe, Springer

Appendix

```
library(tsdl)
tsdl_sales<-subset(tsdl,"Sales")
attributes(tsdl_sales[[1]])
```

```
## $tsp
## [1] 1960.000 1975.917   12.000
##
## $class
## [1] "ts"
##
## $source
## [1] "Abraham & Ledolter (1983)"
##
## $description
## [1] "Monthly gasoline demand Ontario gallon millions 1960 ? 1975"
##
## $subject
## [1] "Sales"
```

```
tsdl_sales[[1]]
```

```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1960  87695  86890  96442  98133 113615 123924 128924 134775 117357 114626
## 1961  92188  88591  98683  99207 125485 124677 132543 140735 124008 121194
## 1962 101007  94228 104255 106922 130621 125251 140318 146174 122318 128770
## 1963 108497 100482 106140 118581 132371 132042 151938 150997 130931 137018
## 1964 109894 106061 112539 125745 136251 140892 158390 148314 144148 140138
## 1965 109895 109044 122499 124264 142296 150693 163331 165837 151731 142491
## 1966 116963 118049 137869 127392 154166 160227 165869 173522 155828 153771
## 1967 124046 121260 138870 129782 162312 167211 172897 189689 166496 160754
## 1968 139625 137361 138963 155301 172026 165004 185861 190270 163903 174270
## 1969 146182 137728 148932 156751 177998 174559 198079 189073 175702 180097
## 1970 154277 144998 159644 168646 166273 190176 205541 193657 182617 189614
## 1971 158167 156261 176353 175720 193939 201269 218960 209861 198688 190474
## 1972 166286 170699 181468 174241 210802 212262 218099 229001 203200 212557
## 1973 188992 175347 196265 203526 227443 233038 234119 255133 216478 232868
## 1974 194784 189756 193522 212870 248565 221532 252642 255007 206826 233231
## 1975 199024 191813 195997 208684 244113 243108 255918 244642 237579 237579
##         Nov    Dec
```

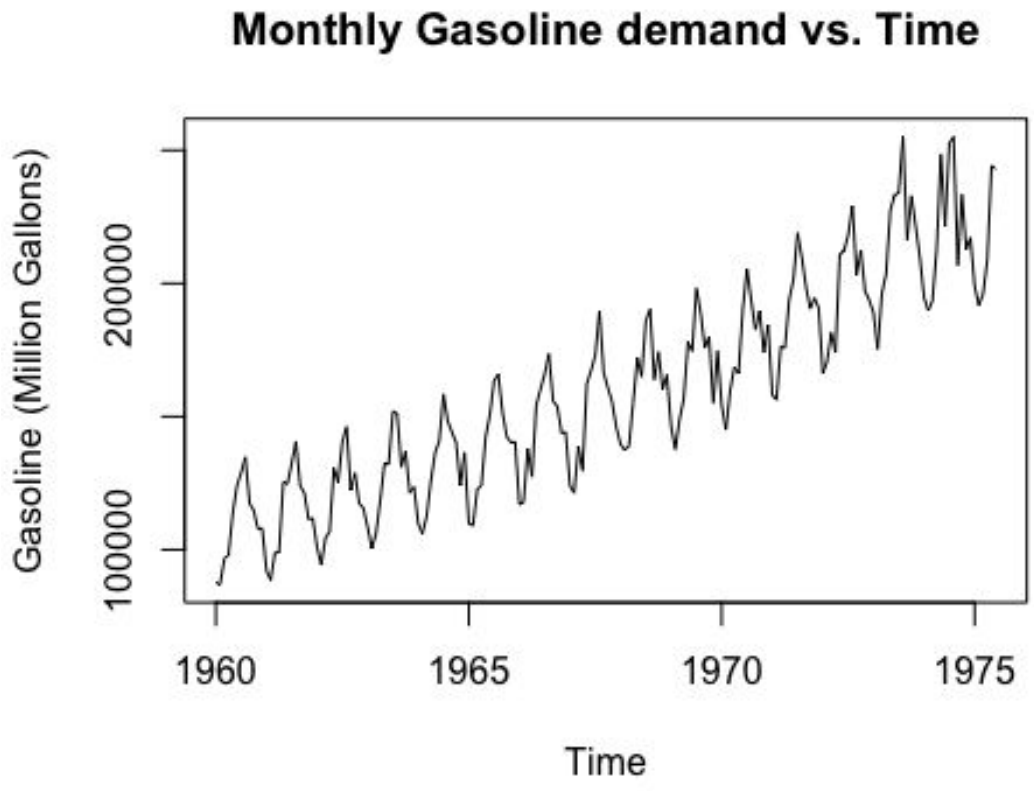
```
## 1960 107677 108087
## 1961 111634 111565
## 1962 117518 115492
## 1963 121271 123548
## 1964 124075 136485
## 1965 140229 140463
## 1966 143963 143898
## 1967 155582 145936
## 1968 160272 165614
## 1969 155202 174508
## 1970 174176 184416
## 1971 194502 190755
## 1972 197095 193693
## 1973 221616 209893
## 1974 212678 217173
## 1975 217775 227621
```

```
original<-tsdl_sales[[1]]
sale <- tsdl_sales[[1]][c(-187:-192)]
#sale<-tsdl_sales[[1]]
sale.ts <- ts(sale, start = c(1960,1), frequency = 12)
sale.ts
```

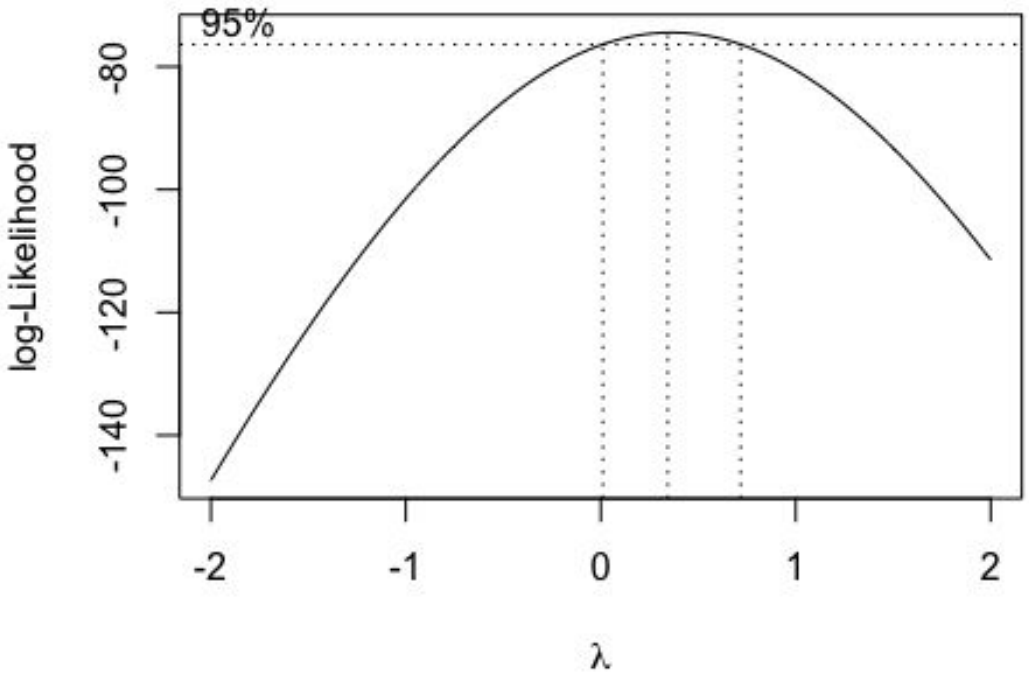
```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1960  87695  86890  96442  98133 113615 123924 128924 134775 117357 114626
## 1961  92188  88591  98683  99207 125485 124677 132543 140735 124008 121194
## 1962 101007  94228 104255 106922 130621 125251 140318 146174 122318 128770
## 1963 108497 100482 106140 118581 132371 132042 151938 150997 130931 137018
## 1964 109894 106061 112539 125745 136251 140892 158390 148314 144148 140138
## 1965 109895 109044 122499 124264 142296 150693 163331 165837 151731 142491
## 1966 116963 118049 137869 127392 154166 160227 165869 173522 155828 153771
## 1967 124046 121260 138870 129782 162312 167211 172897 189689 166496 160754
## 1968 139625 137361 138963 155301 172026 165004 185861 190270 163903 174270
## 1969 146182 137728 148932 156751 177998 174559 198079 189073 175702 180097
## 1970 154277 144998 159644 168646 166273 190176 205541 193657 182617 189614
## 1971 158167 156261 176353 175720 193939 201269 218960 209861 198688 190474
## 1972 166286 170699 181468 174241 210802 212262 218099 229001 203200 212557
## 1973 188992 175347 196265 203526 227443 233038 234119 255133 216478 232868
## 1974 194784 189756 193522 212870 248565 221532 252642 255007 206826 233231
## 1975 199024 191813 195997 208684 244113 243108
##         Nov    Dec
## 1960 107677 108087
## 1961 111634 111565
## 1962 117518 115492
## 1963 121271 123548
## 1964 124075 136485
## 1965 140229 140463
## 1966 143963 143898
```

```
## 1967 155582 145936
## 1968 160272 165614
## 1969 155202 174508
## 1970 174176 184416
## 1971 194502 190755
## 1972 197095 193693
## 1973 221616 209893
## 1974 212678 217173
## 1975
```

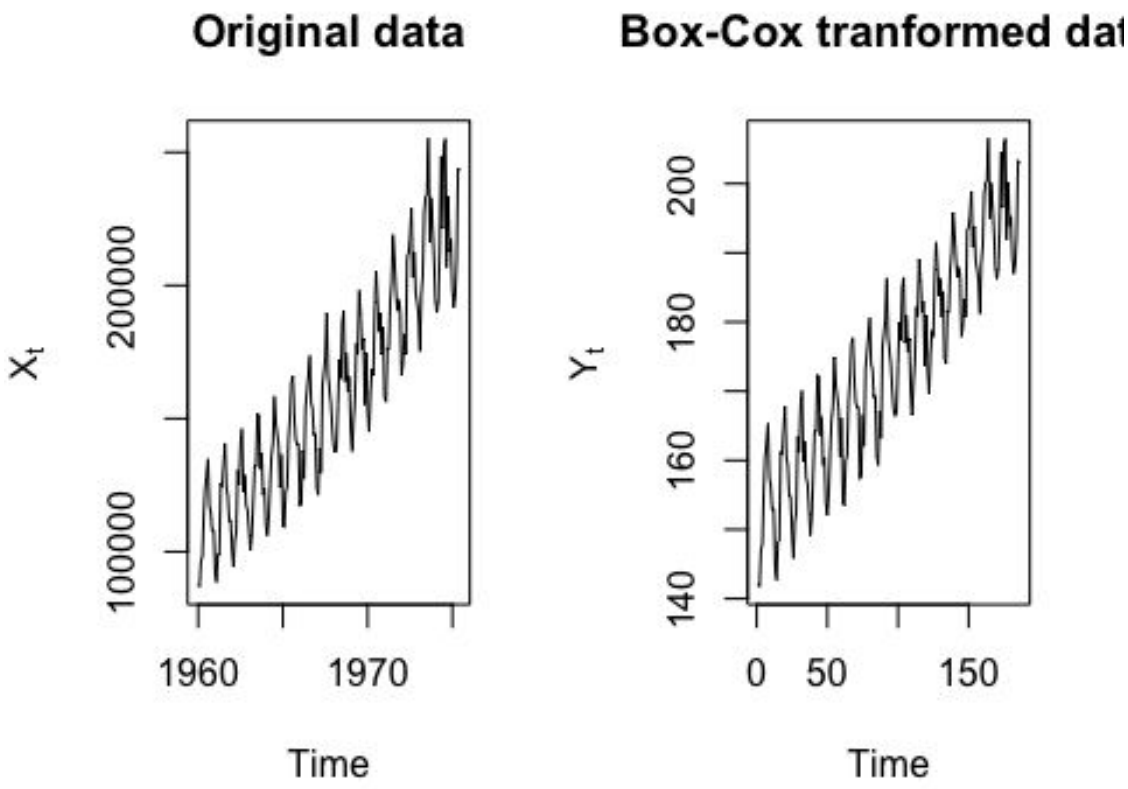
```
ts.plot(sale.ts,ylab = "Gasoline (Million Gallons)",  main = "Monthly
Gasoline demand vs. Time")
```



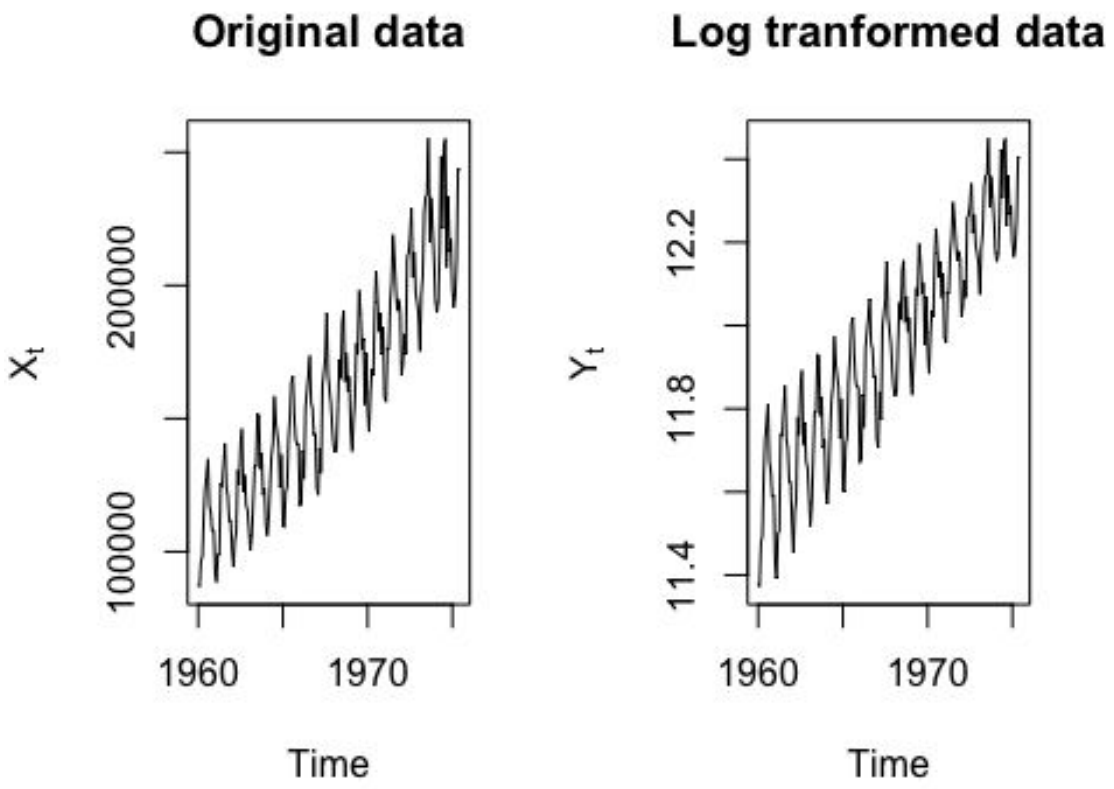

```
library(MASS)
t = 1:length(sale.ts)
fit = lm(sale.ts ~ t)
bcTransform = boxcox(sale.ts ~ t, plotit = TRUE)
```




```
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
sale.bc = (1/lambda)*(sale^lambda-1)
sale.log = log(sale.ts)
sale.sqrt = sqrt(sale.ts)
op <- par(mfrow = c(1,2))
ts.plot(sale.ts,main = "Original data",ylab = expression(X[t]))
ts.plot(sale.bc,main = "Box-Cox tranformed data", ylab = expression(Y[t]))
```




```r
ts.plot(sale.ts,main = "Original data",ylab = expression(X[t]))
ts.plot(sale.log,main = "Log tranformed data", ylab = expression(Y[t]))
```




```
var(sale.ts)
```

```
## [1] 1600021030
```

```
var(sale.bc)
```

```
## [1] 236.3724
```

```
var(sale.log)
```

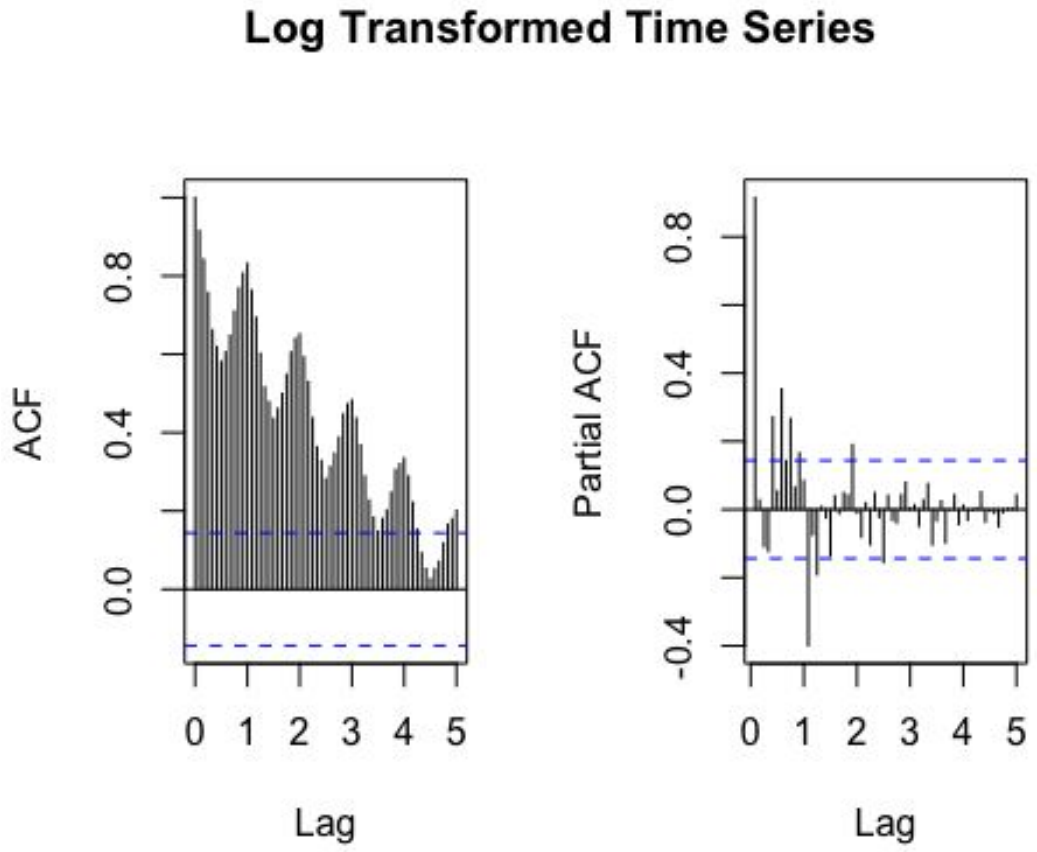
```
## [1] 0.06465378
```

```
var(sale.sqrt)
```

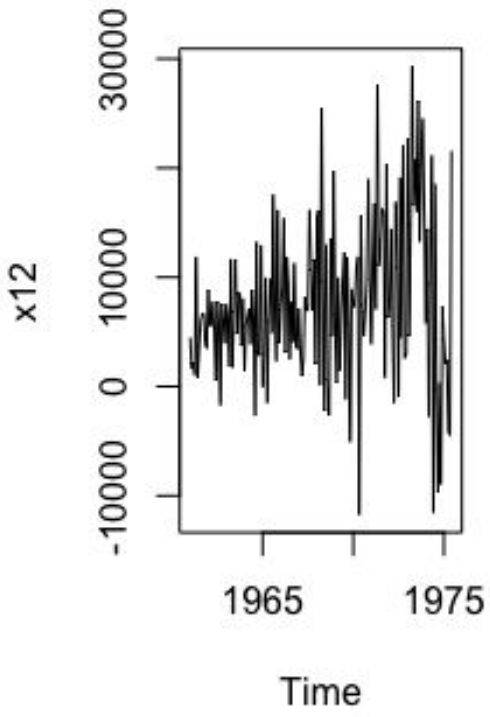
```
## [1] 2501.043
```

Log transformation has the lowest variance, so we go with log transformation.

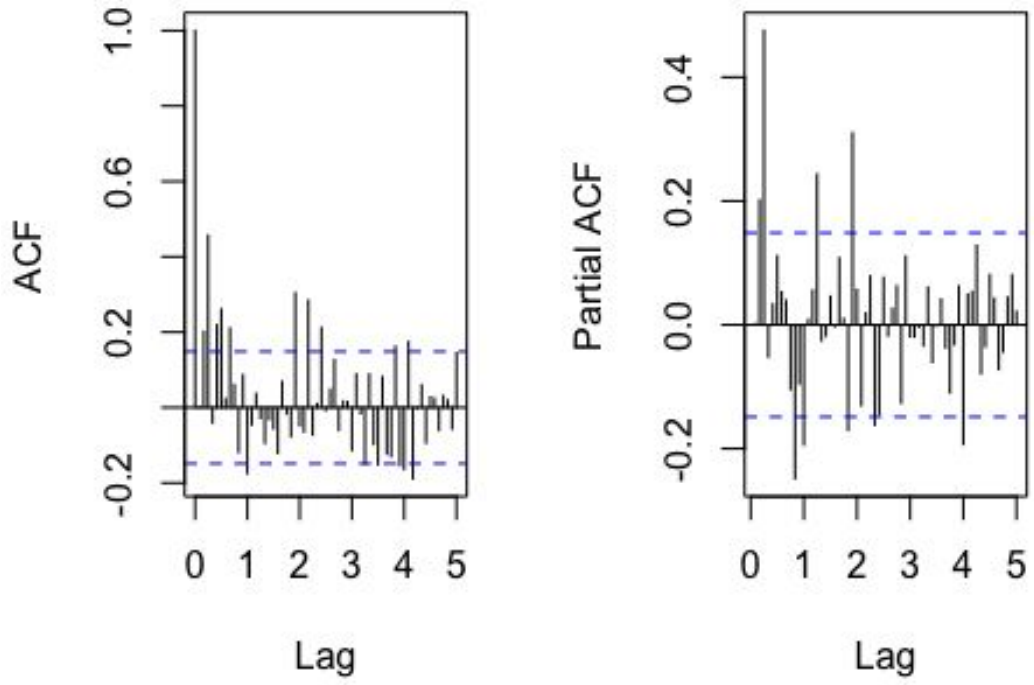
```
op = par(mfrow = c(1,2))
acf(sale.log,lag.max = 60,main = "")
pacf(sale.log,lag.max = 60,main = "")
title("Log Transformed Time Series", line = -1, outer=TRUE)
```




```
x12 = diff(sale.ts, 12)
var(x12)

## [1] 53433643

ts.plot(x12)
op = par(mfrow = c(1,2))
```



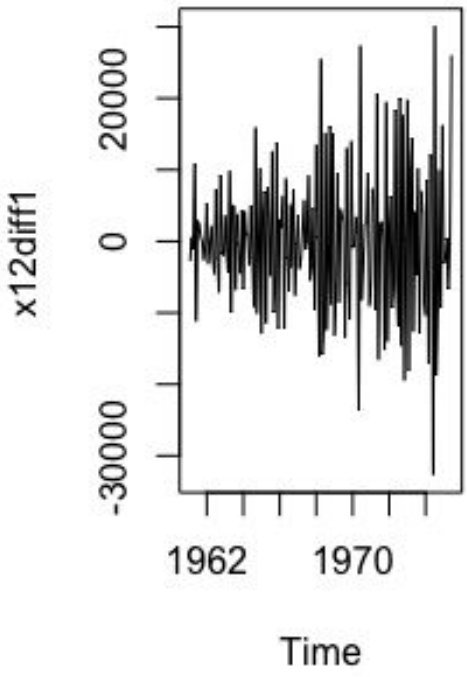

```
acf(x12,lag.max = 60,main = "")
pacf(x12,lag.max = 60,main = "")
title("Differenced at 12", line = -1, outer=TRUE)
```



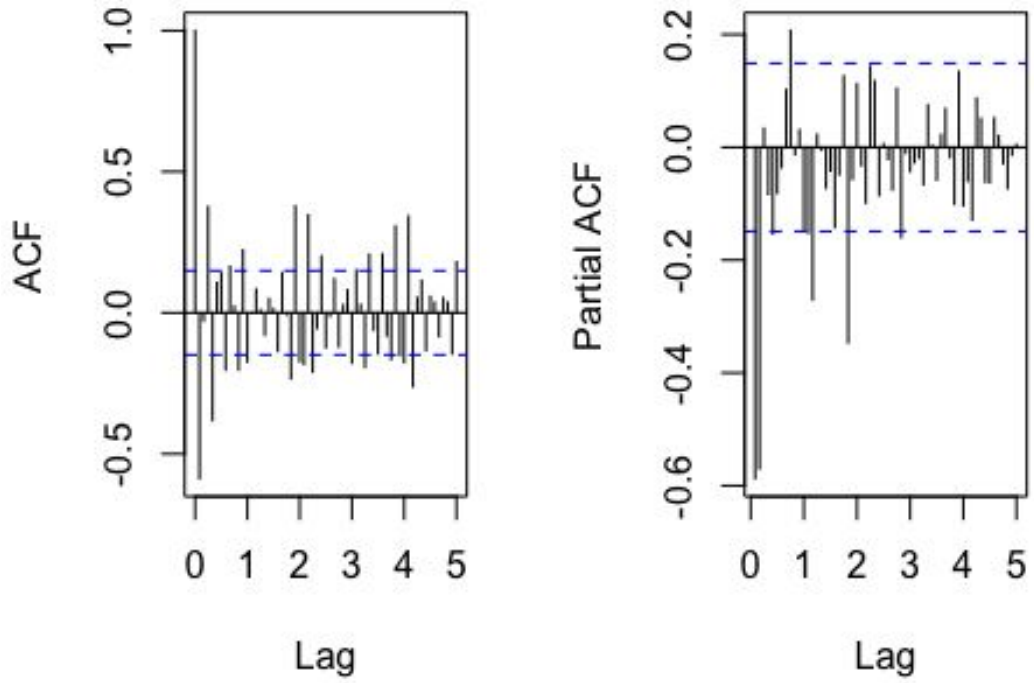

```
x12diff1 = diff(x12, 1)
var(x12diff1)
```

```
## [1] 106149321
```

```
ts.plot(x12diff1)
op = par(mfrow = c(1,2))
```



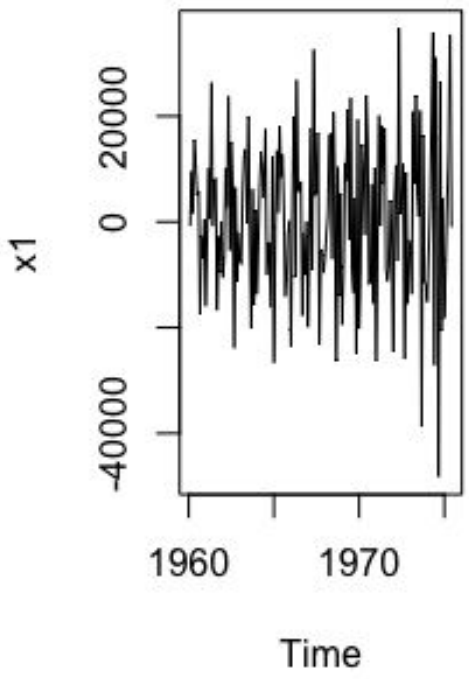

```
acf(x12diff1,lag.max = 60,main = "")
pacf(x12diff1,lag.max = 60,main = "")
title("x12diff1", line = -1, outer=TRUE)
```



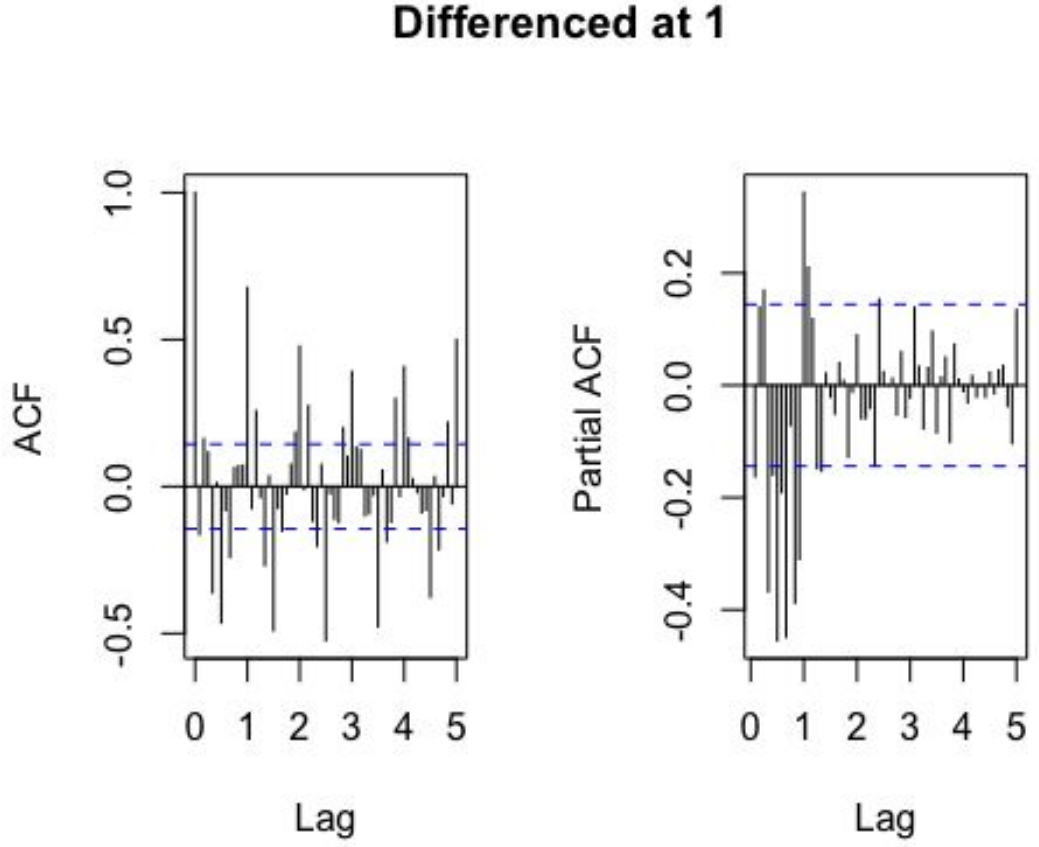

```
x1 = diff(sale.ts, 1)
var(x1)
```

```
## [1] 214321914
```

```
ts.plot(x1)
op = par(mfrow = c(1,2))
```



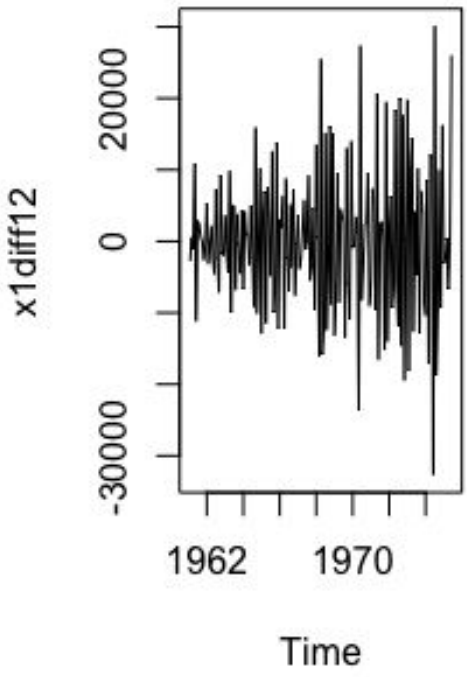

```
acf(x1,lag.max = 60,main = "")
pacf(x1,lag.max = 60,main = "")
title("Differenced at 1", line = -1, outer=TRUE)
```



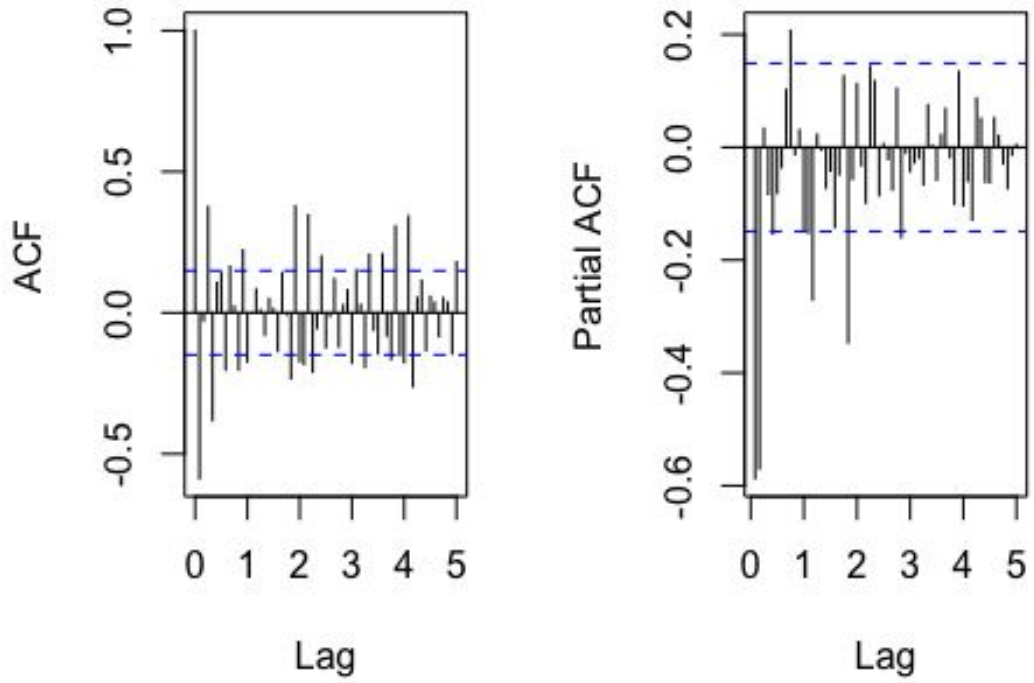

```
x1diff12 = diff(x1, 12)
var(x1diff12)
```

```
## [1] 106149321
```

```
ts.plot(x1diff12)
op = par(mfrow = c(1,2))
```



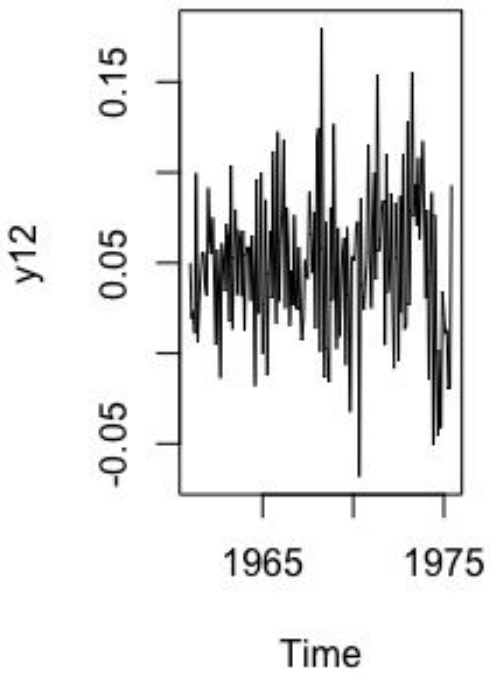

```
acf(x1diff12,lag.max = 60,main = "")
pacf(x1diff12,lag.max = 60,main = "")
title("x1diff12", line = -1, outer=TRUE)
```



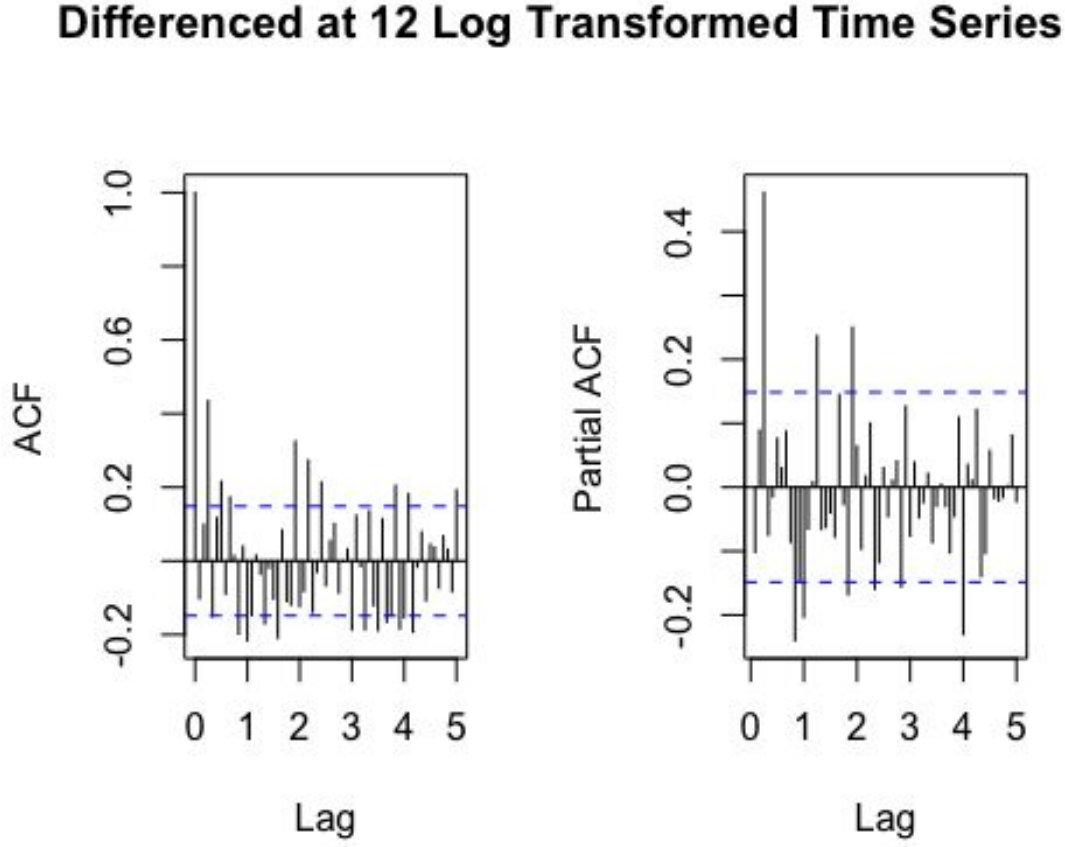

```
y12 = diff(sale.log, 12)
var(y12)
```

```
## [1] 0.001672437
```

```
ts.plot(y12)
op = par(mfrow = c(1,2))
```



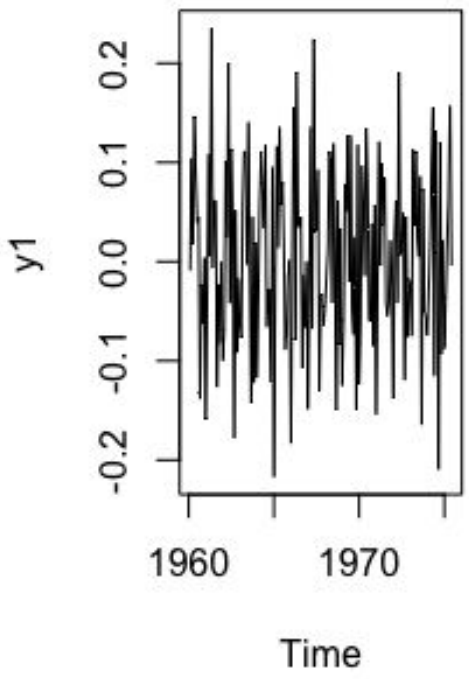

```
acf(y12,lag.max = 60,main = "")
pacf(y12,lag.max = 60,main = "")
title("Differenced at 12 Log Transformed Time Series", line = -1, outer=TRUE)
```



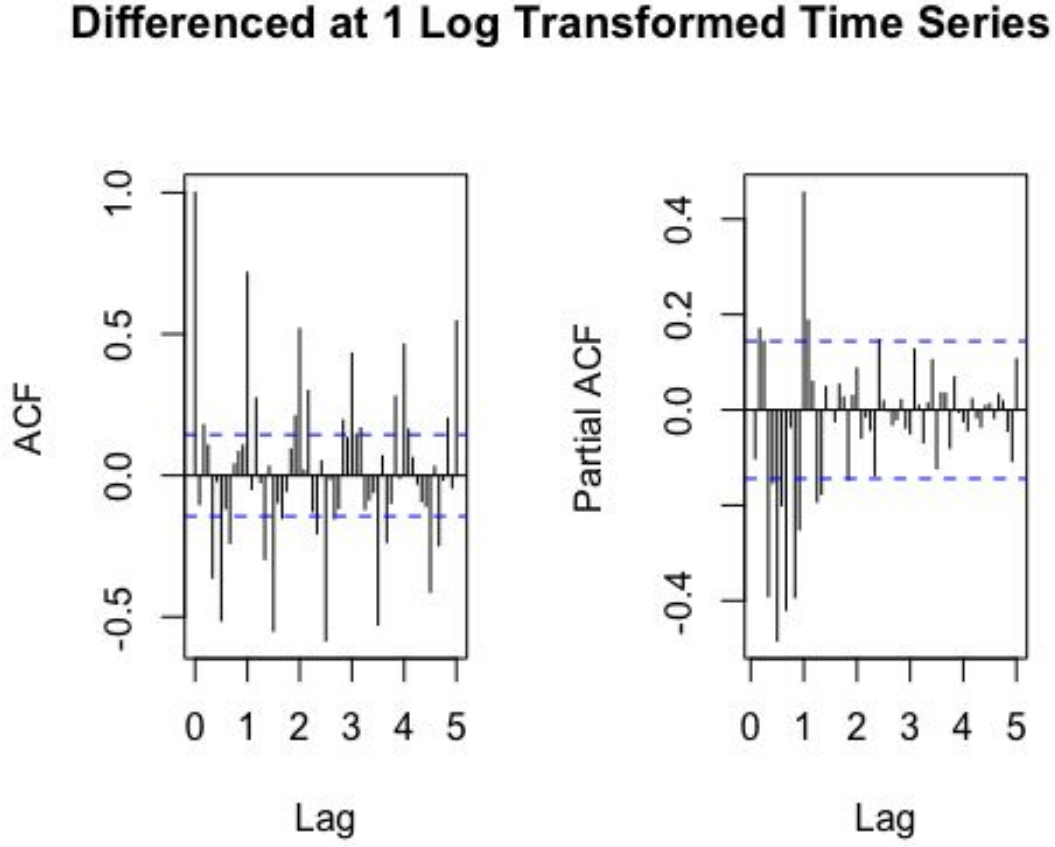

```
y1 = diff(sale.log, 1)
var(y1)
```

```
## [1] 0.008146013
```

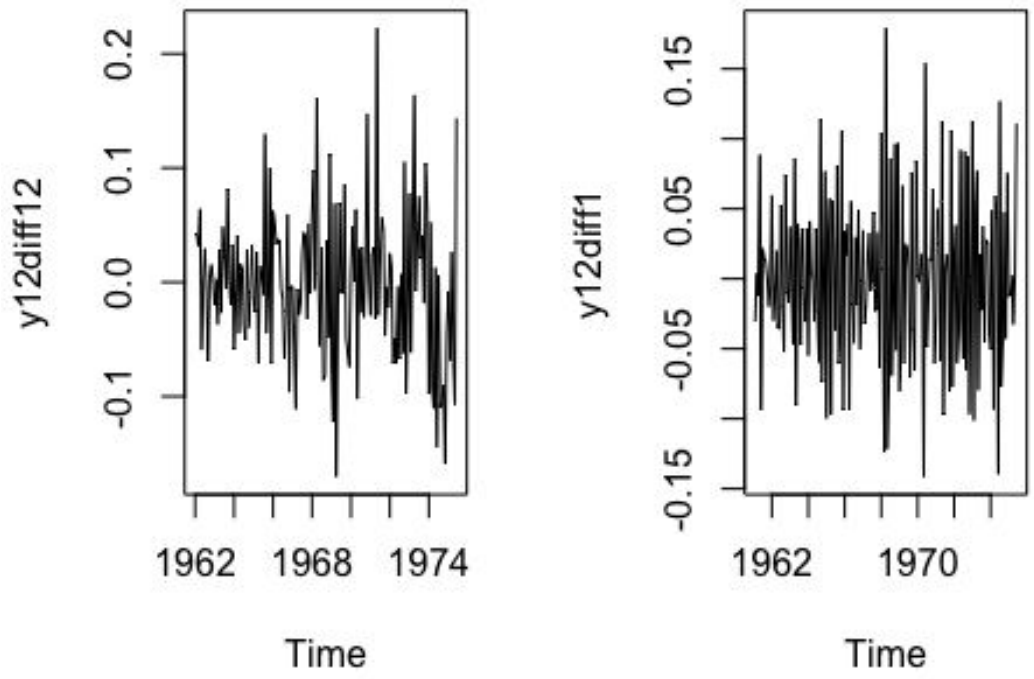
```
ts.plot(y1)
op = par(mfrow = c(1,2))
```



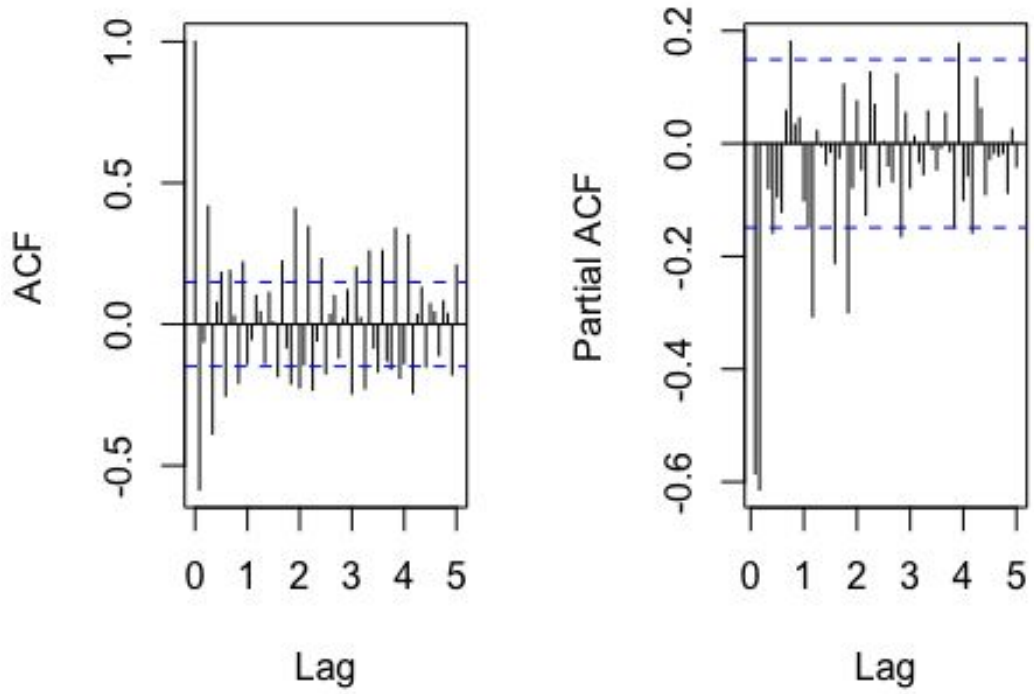

```
acf(y1,lag.max = 60,main = "")
pacf(y1,lag.max = 60,main = "")
title("Differenced at 1 Log Transformed Time Series", line = -1, outer=TRUE)
```




```
y12diff12 = diff(y12, 12)
var(y12diff12)

## [1] 0.004083666

ts.plot(y12diff12)

y12diff1 = diff(y12, 1)
var(y12diff1)

## [1] 0.003695186

ts.plot(y12diff1)
```




```r
op = par(mfrow = c(1,2))
acf(y12diff1,lag.max = 60,main = "")
pacf(y12diff1,lag.max = 60,main = "")
title("y12diff1", line = -1, outer=TRUE)
```




```
y1diff12 = diff(y1, 12)
var(y1diff12)
```

```
## [1] 0.003695186
```

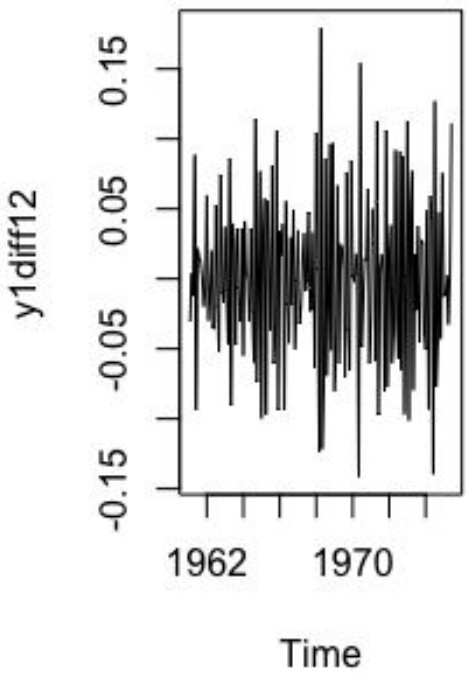
```
a1 = diff(y1diff12, 1)
var(a1)
```

```
## [1] 0.01170392
```

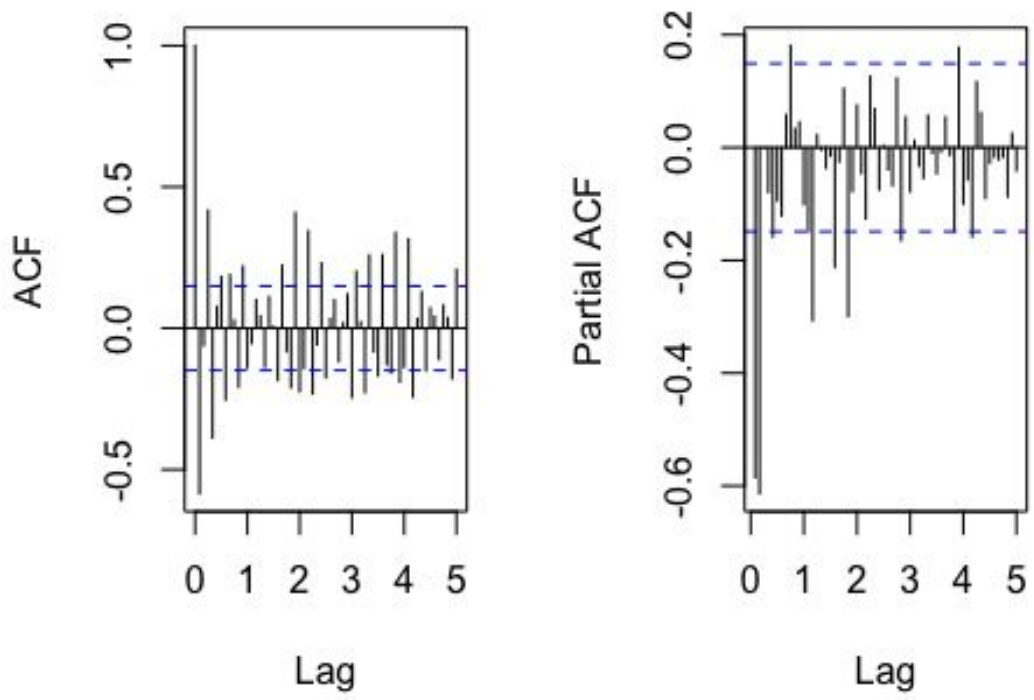
```
a12 = diff(y1diff12, 12)
var(a12)
```

```
## [1] 0.00861073
```

```
ts.plot(y1diff12)
op = par(mfrow = c(1,2))
```



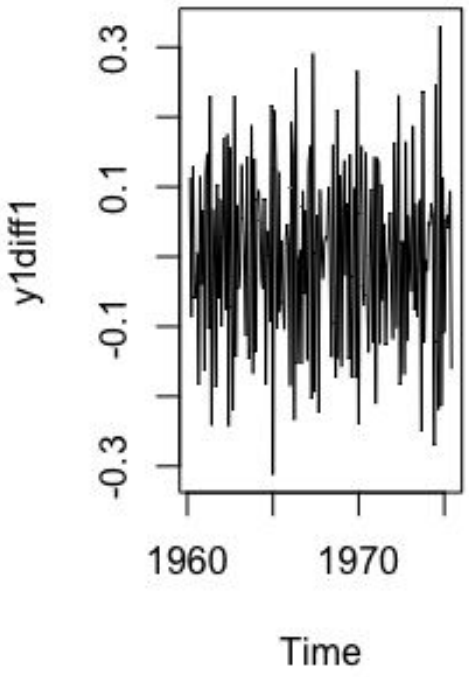

```
acf(y1diff12,lag.max = 60,main = "")
pacf(y1diff12,lag.max = 60,main = "")
title("y1diff12", line = -1, outer=TRUE)
```



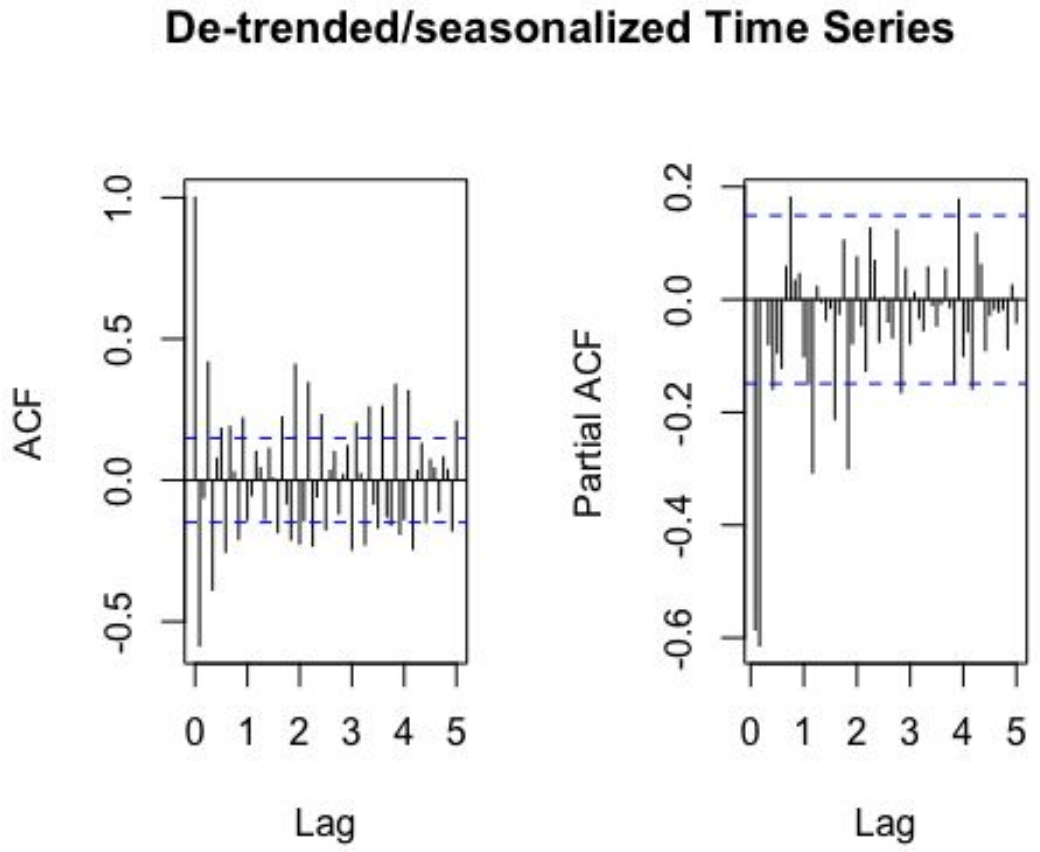

```
y1diff1 = diff(y1, 1)
var(y1diff1)
```

```
## [1] 0.01803993
```

```
ts.plot(y1diff1)
```



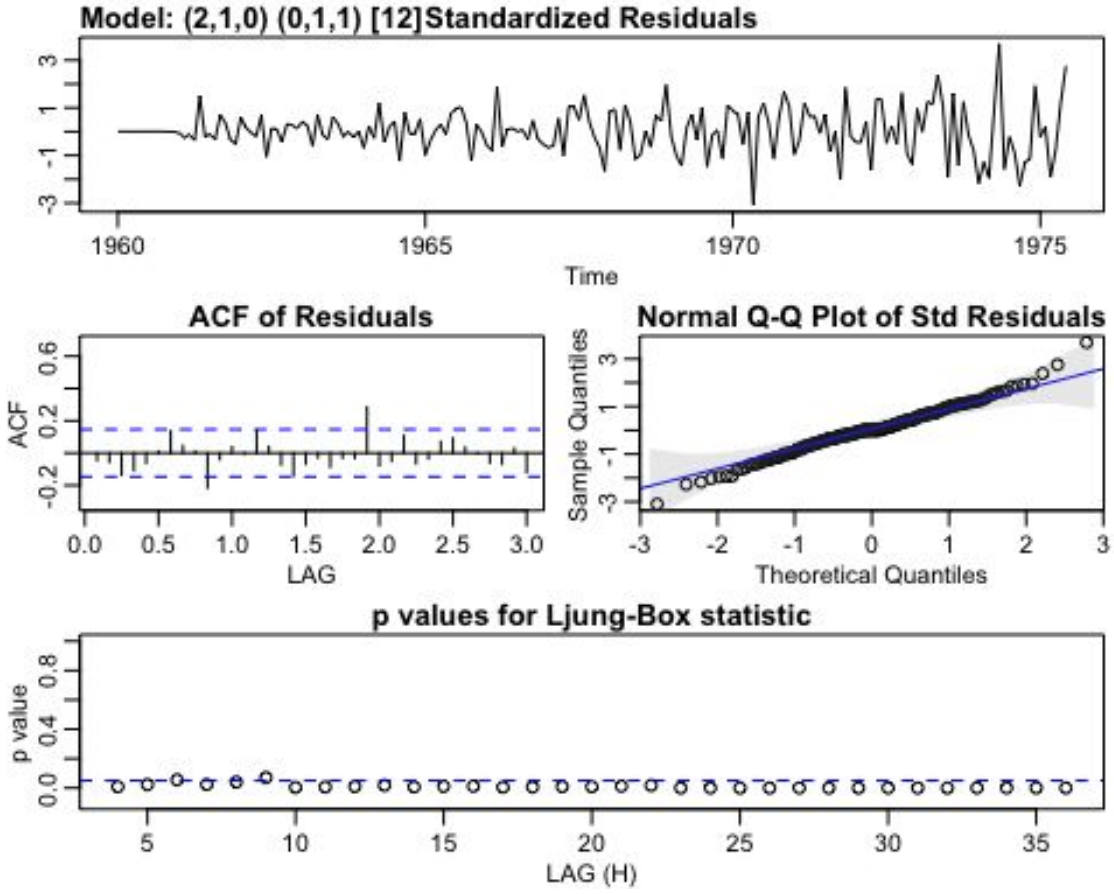

```
op = par(mfrow = c(1,2))
acf(y1diff12,lag.max = 60,main = "")
pacf(y1diff12,lag.max = 60,main = "")
title("De-trended/seasonalized Time Series", line = -1, outer=TRUE)
```



ACF cuts off after lag 1 and possibly lag 2 for every 12 ticks while the PACF tails off leading me to believe that the model follows a (0,1,1) or a (0,1,2) seasonal component. For the non seasonal part, PACF cuts off after lag = 2 and lag = 5 meaning that it probably has an AR(2) or AR(5) counterpart.

```
library(astsa)

## Warning: package 'astsa' was built under R version 3.5.2

m1<-sarima(sale.ts,2,1,0,0,1,1, 12)

## initial  value 9.242919
## iter   2 value 8.979267
## iter   3 value 8.870460
## iter   4 value 8.664616
## iter   5 value 8.652544
## iter   6 value 8.651988
## iter   7 value 8.650959
## iter   8 value 8.650873
## iter   9 value 8.650859
## iter  10 value 8.650857
## iter  11 value 8.650857
## iter  11 value 8.650857
```

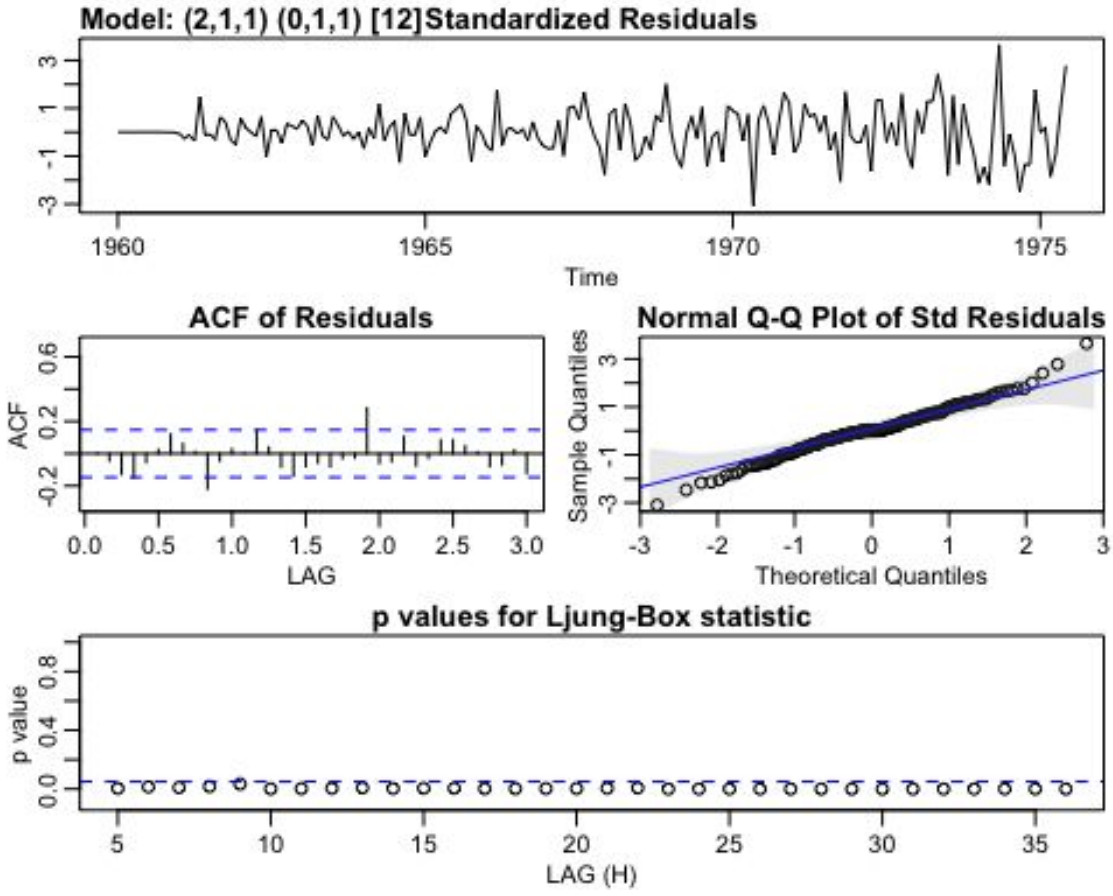
```
## iter  11 value 8.650857
## final  value 8.650857
## converged
## initial  value 8.664921
## iter   2 value 8.664127
## iter   3 value 8.663878
## iter   4 value 8.663706
## iter   4 value 8.663706
## iter   4 value 8.663706
## final  value 8.663706
## converged
```




```
m2<-sarima(sale.ts,2,1,1,0,1,1, 12)
```

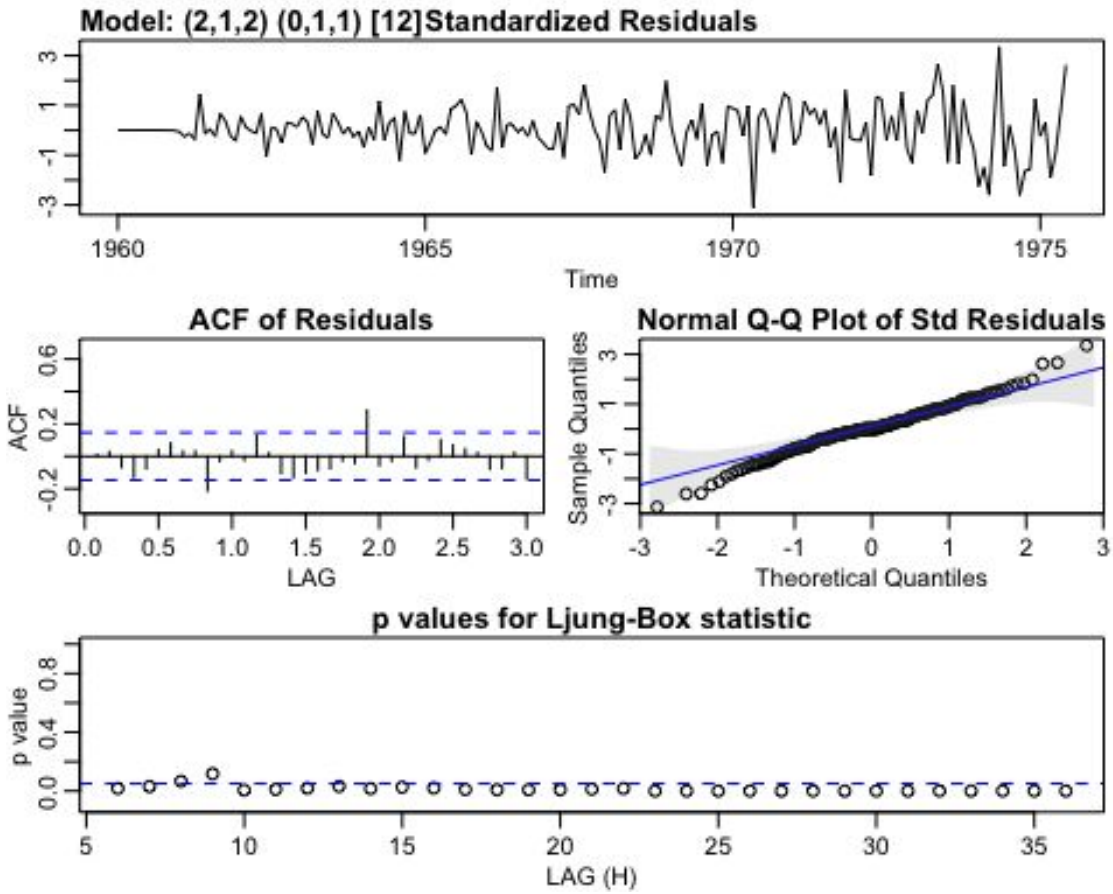
```
## initial  value 9.242919
## iter   2 value 8.811148
## iter   3 value 8.774551
## iter   4 value 8.691332
## iter   5 value 8.674981
## iter   6 value 8.666453
## iter   7 value 8.658580
## iter   8 value 8.649419
```

```
## iter   9 value 8.648761
## iter  10 value 8.648697
## iter  11 value 8.648695
## iter  12 value 8.648693
## iter  13 value 8.648693
## iter  13 value 8.648692
## iter  13 value 8.648692
## final  value 8.648692
## converged
## initial  value 8.662514
## iter   2 value 8.661749
## iter   3 value 8.661493
## iter   4 value 8.661318
## iter   5 value 8.661317
## iter   6 value 8.661316
## iter   7 value 8.661316
## iter   7 value 8.661316
## iter   7 value 8.661316
## final  value 8.661316
## converged
```



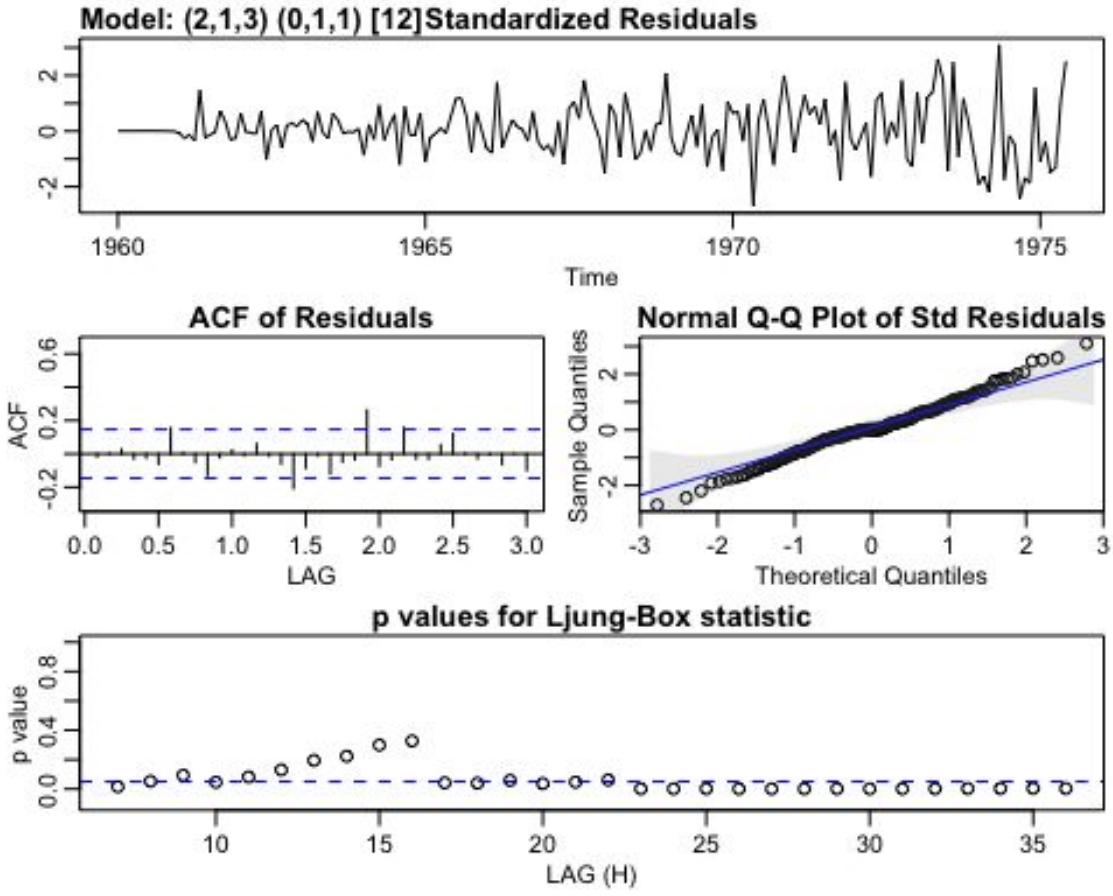
```
m3<-sarima(sale.ts,2,1,2,0,1,1, 12)
```

```
## initial  value 9.242919
## iter   2 value 8.809413
## iter   3 value 8.776536
## iter   4 value 8.696937
## iter   5 value 8.675785
## iter   6 value 8.669517
## iter   7 value 8.660686
## iter   8 value 8.655676
## iter   9 value 8.648205
## iter  10 value 8.644756
## iter  11 value 8.643481
## iter  12 value 8.643092
## iter  13 value 8.643078
## iter  14 value 8.643026
## iter  15 value 8.643023
## iter  16 value 8.643022
## iter  16 value 8.643022
## iter  16 value 8.643022
## final  value 8.643022
## converged
## initial  value 8.656464
## iter   2 value 8.655700
## iter   3 value 8.655519
## iter   4 value 8.655409
## iter   5 value 8.655407
## iter   6 value 8.655406
## iter   7 value 8.655406
## iter   7 value 8.655406
## iter   7 value 8.655406
## final  value 8.655406
## converged
```



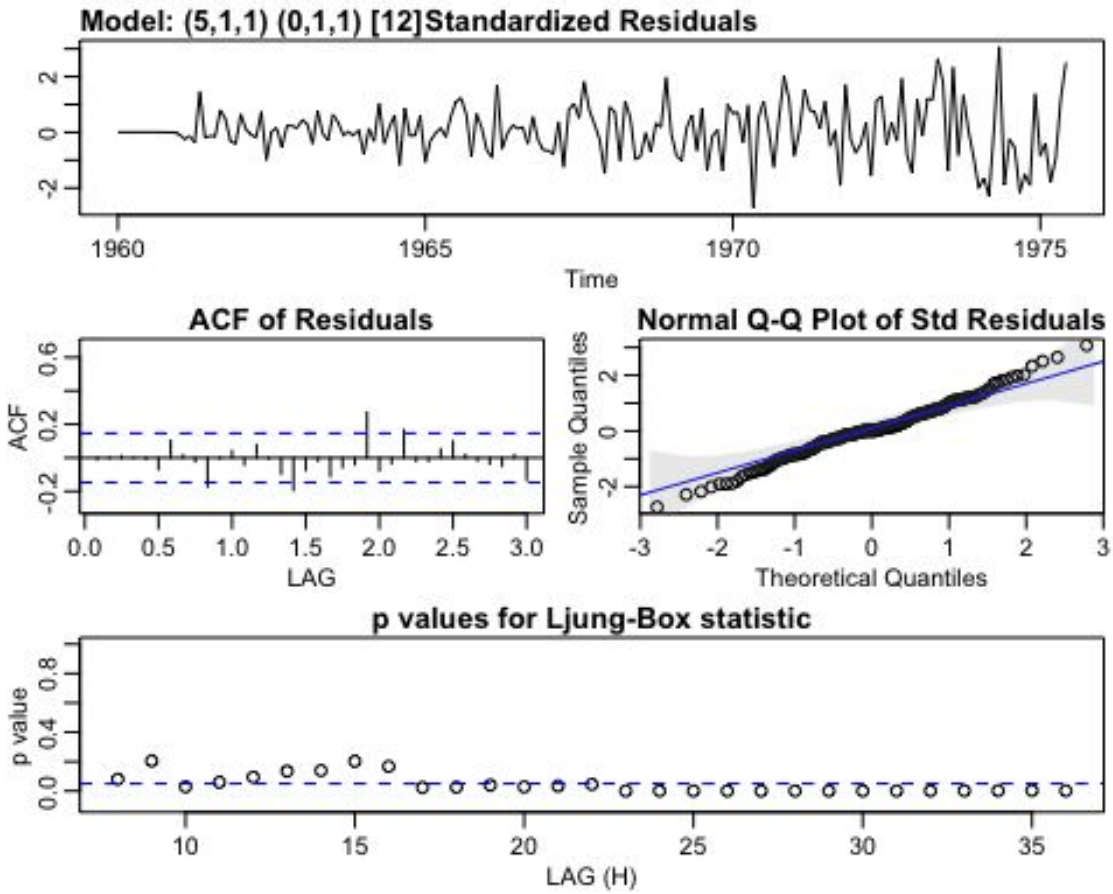

```
m4<-sarima(sale.ts,2,1,3,0,1,1, 12)

## initial  value 9.242919
## iter   2 value 8.904305
## iter   3 value 8.794421
## iter   4 value 8.765189
## iter   5 value 8.742318
## iter   6 value 8.707570
## iter   7 value 8.698749
## iter   8 value 8.690864
## iter   9 value 8.671467
## iter  10 value 8.658112
## iter  11 value 8.641880
## iter  12 value 8.635329
## iter  13 value 8.628157
## iter  14 value 8.626193
## iter  15 value 8.625709
## iter  16 value 8.625274
## iter  17 value 8.624917
## iter  18 value 8.624733
## iter  19 value 8.624687
## iter  20 value 8.624673
```

```
## iter  21 value 8.624657
## iter  22 value 8.624654
## iter  23 value 8.624654
## iter  23 value 8.624654
## iter  23 value 8.624654
## final  value 8.624654
## converged
## initial  value 8.636893
## iter   2 value 8.636600
## iter   3 value 8.636598
## iter   4 value 8.636179
## iter   5 value 8.636176
## iter   6 value 8.636174
## iter   7 value 8.636169
## iter   8 value 8.636168
## iter   9 value 8.636166
## iter  10 value 8.636166
## iter  10 value 8.636166
## iter  10 value 8.636166
## final  value 8.636166
## converged
```



```
m5<-sarima(sale.ts,5,1,1,0,1,1, 12)
```

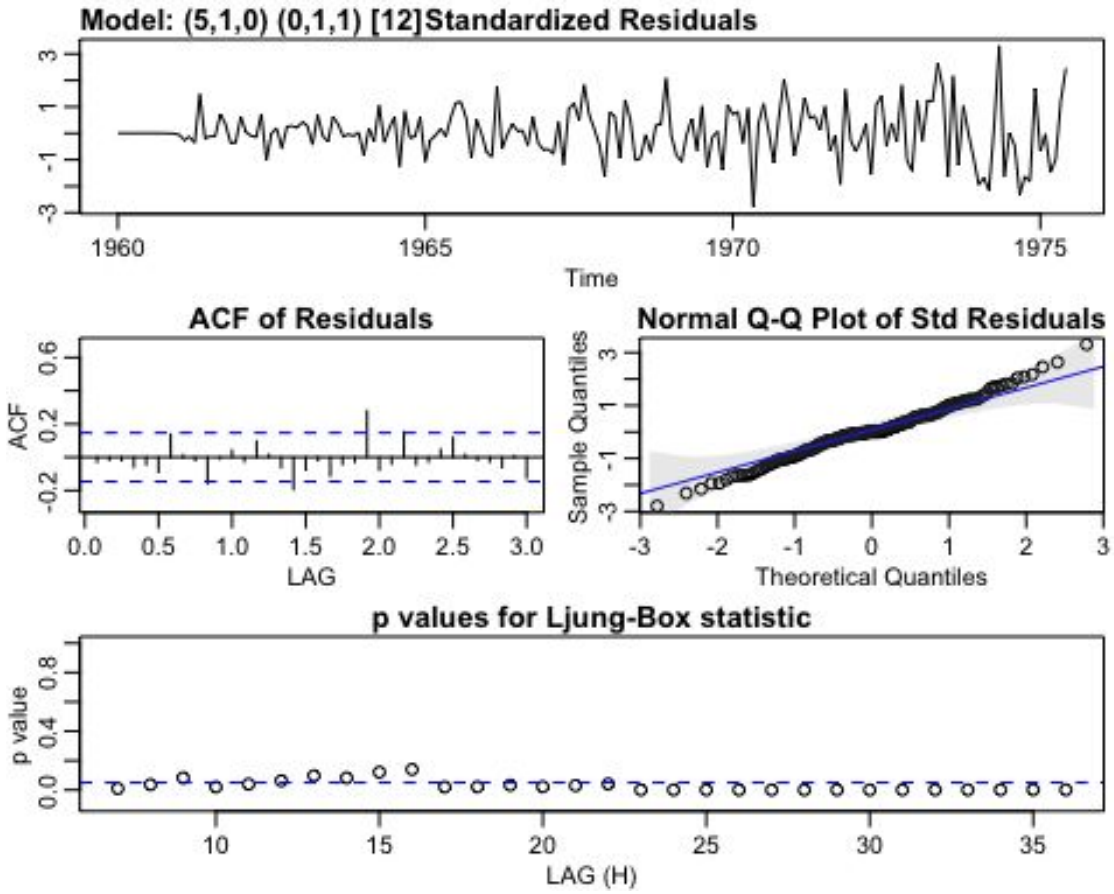
```
## initial  value 9.245108
## iter   2 value 8.892412
## iter   3 value 8.820707
## iter   4 value 8.718158
## iter   5 value 8.653136
## iter   6 value 8.626022
## iter   7 value 8.624432
## iter   8 value 8.624110
## iter   9 value 8.623787
## iter  10 value 8.623326
## iter  11 value 8.622429
## iter  12 value 8.621467
## iter  13 value 8.620835
## iter  14 value 8.620741
## iter  15 value 8.620734
## iter  16 value 8.620730
## iter  17 value 8.620729
## iter  18 value 8.620728
## iter  19 value 8.620727
## iter  20 value 8.620727
## iter  20 value 8.620727
## iter  20 value 8.620727
## final  value 8.620727
## converged
## initial  value 8.639007
## iter   2 value 8.638162
## iter   3 value 8.637716
## iter   4 value 8.636470
## iter   5 value 8.636449
## iter   6 value 8.636434
## iter   7 value 8.636433
## iter   8 value 8.636426
## iter   9 value 8.636424
## iter  10 value 8.636422
## iter  11 value 8.636421
## iter  11 value 8.636421
## iter  11 value 8.636421
## final  value 8.636421
## converged
```




```
m6<-sarima(sale.ts,5,1,0,0,1,1, 12)
```

```
## initial  value 9.245108
## iter   2 value 9.089518
## iter   3 value 8.797602
## iter   4 value 8.763191
## iter   5 value 8.692300
## iter   6 value 8.660326
## iter   7 value 8.642462
## iter   8 value 8.636139
## iter   9 value 8.629141
## iter  10 value 8.627825
## iter  11 value 8.627560
## iter  12 value 8.627460
## iter  13 value 8.627459
## iter  14 value 8.627459
## iter  14 value 8.627459
## final  value 8.627459
## converged
## initial  value 8.647239
## iter   2 value 8.645622
## iter   3 value 8.644653
```

```
## iter   4 value 8.643947
## iter   5 value 8.643939
## iter   6 value 8.643938
## iter   6 value 8.643938
## iter   6 value 8.643938
## final  value 8.643938
## converged
```

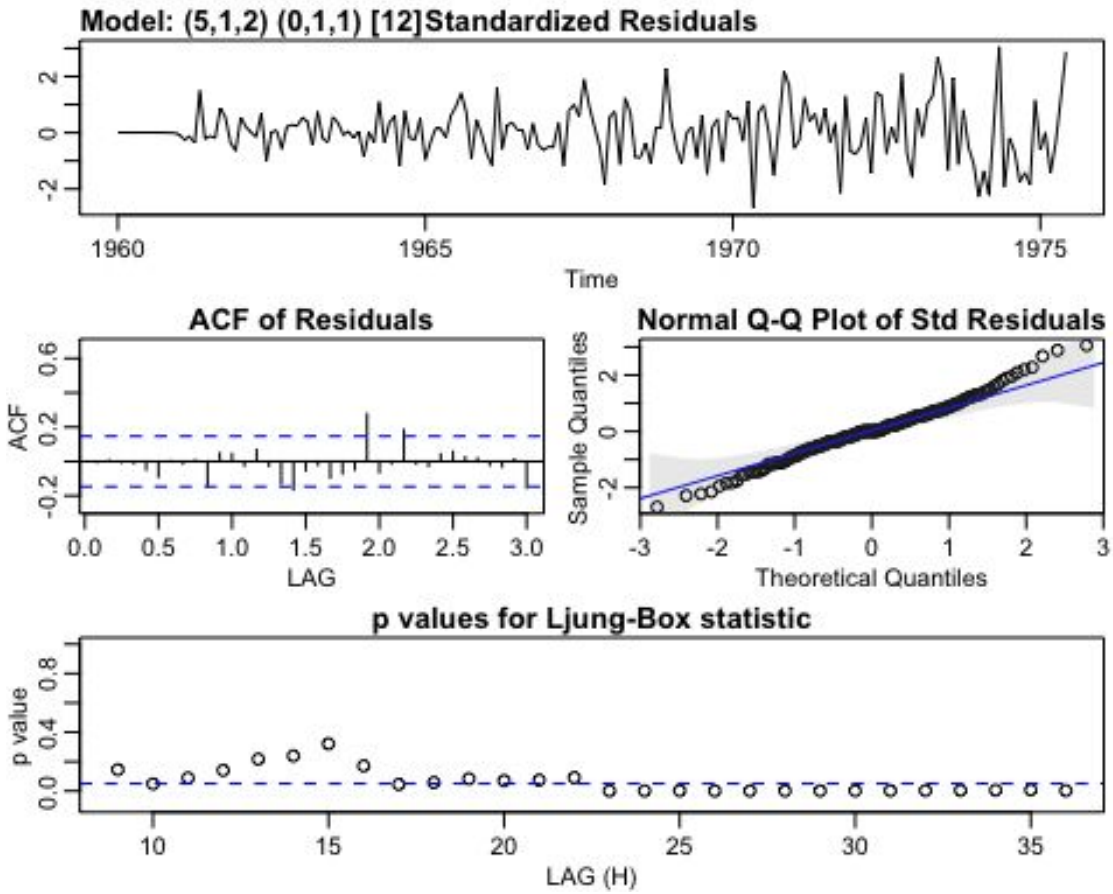



```
m7<-sarima(sale.ts,5,1,2,0,1,1, 12)
```

```
## initial  value 9.245108
## iter   2 value 8.889246
## iter   3 value 8.818575
## iter   4 value 8.702024
## iter   5 value 8.661793
## iter   6 value 8.626109
## iter   7 value 8.623407
## iter   8 value 8.622672
## iter   9 value 8.622290
## iter  10 value 8.622021
## iter  11 value 8.621465
## iter  12 value 8.620573
```

```
## iter  13 value 8.619784
## iter  14 value 8.619433
## iter  15 value 8.619387
## iter  16 value 8.619383
## iter  17 value 8.619381
## iter  18 value 8.619380
## iter  19 value 8.619379
## iter  20 value 8.619378
## iter  21 value 8.619376
## iter  22 value 8.619373
## iter  23 value 8.619366
## iter  24 value 8.619360
## iter  25 value 8.619359
## iter  25 value 8.619359
## iter  25 value 8.619359
## final  value 8.619359
## converged
## initial  value 8.643384
## iter   2 value 8.639764
## iter   3 value 8.638980
## iter   4 value 8.638377
## iter   5 value 8.638319
## iter   6 value 8.638132
## iter   7 value 8.637544
## iter   8 value 8.637226
## iter   9 value 8.636899
## iter  10 value 8.636786
## iter  11 value 8.636724
## iter  12 value 8.636583
## iter  13 value 8.636150
## iter  14 value 8.635985
## iter  15 value 8.635693
## iter  16 value 8.635237
## iter  17 value 8.633355
## iter  18 value 8.633025
## iter  19 value 8.631818
## iter  20 value 8.631196
## iter  21 value 8.630432
## iter  22 value 8.629508
## iter  23 value 8.629167
## iter  24 value 8.628099
## iter  25 value 8.627416
## iter  26 value 8.626868
## iter  27 value 8.626287
## iter  28 value 8.625706
## iter  29 value 8.625351
## iter  30 value 8.625157
## iter  31 value 8.624879
```

```
## iter  32 value 8.624770
## iter  33 value 8.624712
## iter  34 value 8.624674
## iter  35 value 8.624650
## iter  36 value 8.624649
## iter  37 value 8.624648
## iter  38 value 8.624648
## iter  39 value 8.624647
## iter  40 value 8.624646
## iter  41 value 8.624644
## iter  42 value 8.624637
## iter  43 value 8.624621
## iter  44 value 8.624613
## iter  45 value 8.624606
## iter  46 value 8.624543
## iter  47 value 8.624417
## iter  48 value 8.624325
## iter  49 value 8.624303
## iter  50 value 8.624284
## iter  51 value 8.624268
## iter  52 value 8.624254
## iter  53 value 8.624253
## iter  54 value 8.624249
## iter  55 value 8.624239
## iter  56 value 8.624238
## iter  57 value 8.624237
## iter  58 value 8.624235
## iter  59 value 8.624233
## iter  60 value 8.624229
## iter  61 value 8.624226
## iter  62 value 8.624225
## iter  63 value 8.624225
## iter  63 value 8.624225
## iter  63 value 8.624225
## final  value 8.624225
## converged
```




```
m8<-sarima(sale.ts,5,1,3,0,1,1, 12)
```

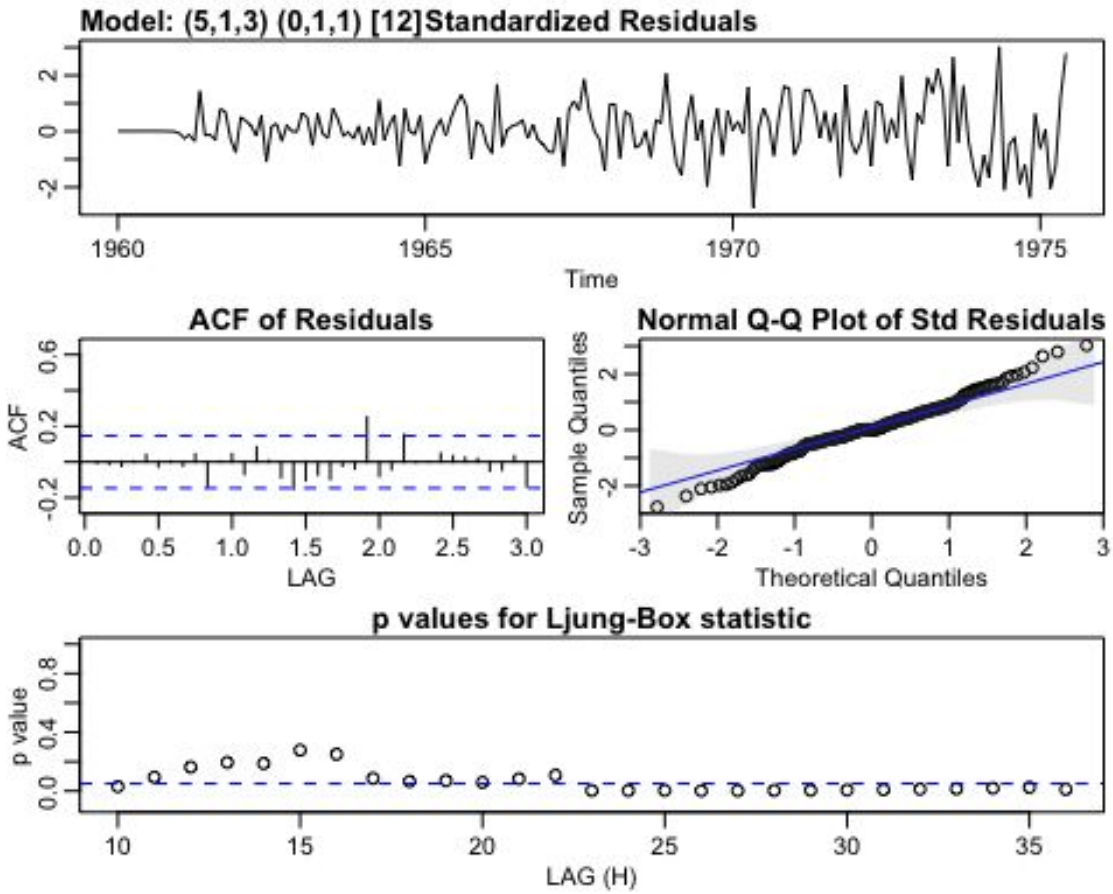
```
## initial  value 9.245108
## iter   2 value 8.986826
## iter   3 value 8.868842
## iter   4 value 8.757114
## iter   5 value 8.679529
## iter   6 value 8.637377
## iter   7 value 8.627718
## iter   8 value 8.626479
## iter   9 value 8.625774
## iter  10 value 8.625042
## iter  11 value 8.623214
## iter  12 value 8.622128
## iter  13 value 8.620497
## iter  14 value 8.619631
## iter  15 value 8.618989
## iter  16 value 8.618955
## iter  17 value 8.618944
## iter  18 value 8.618916
## iter  19 value 8.618846
## iter  20 value 8.618772
```

```
## iter  21 value 8.618765
## iter  22 value 8.618742
## iter  23 value 8.618729
## iter  24 value 8.618724
## iter  25 value 8.618719
## iter  26 value 8.618713
## iter  27 value 8.618710
## iter  28 value 8.618696
## iter  29 value 8.618690
## iter  30 value 8.618687
## iter  31 value 8.618686
## iter  32 value 8.618683
## iter  33 value 8.618677
## iter  34 value 8.618670
## iter  35 value 8.618666
## iter  36 value 8.618664
## iter  37 value 8.618663
## iter  38 value 8.618659
## iter  39 value 8.618650
## iter  40 value 8.618646
## iter  41 value 8.618644
## iter  42 value 8.618643
## iter  43 value 8.618642
## iter  44 value 8.618641
## iter  45 value 8.618639
## iter  46 value 8.618636
## iter  47 value 8.618629
## iter  48 value 8.618626
## iter  49 value 8.618624
## iter  50 value 8.618624
## iter  51 value 8.618623
## iter  52 value 8.618620
## iter  53 value 8.618614
## iter  54 value 8.618601
## iter  55 value 8.618570
## iter  56 value 8.618567
## iter  57 value 8.618561
## iter  58 value 8.618549
## iter  59 value 8.618544
## iter  60 value 8.618535
## iter  61 value 8.618529
## iter  62 value 8.618527
## iter  63 value 8.618526
## iter  64 value 8.618525
## iter  65 value 8.618522
## iter  66 value 8.618516
## iter  67 value 8.618500
## iter  68 value 8.618416
```

```
## iter  69 value 8.618403
## iter  70 value 8.618383
## iter  71 value 8.618371
## iter  72 value 8.618348
## iter  73 value 8.618331
## iter  74 value 8.618276
## iter  75 value 8.618045
## iter  76 value 8.617715
## iter  77 value 8.616973
## iter  78 value 8.616858
## iter  79 value 8.616213
## iter  80 value 8.615332
## iter  81 value 8.611782
## iter  82 value 8.608168
## iter  83 value 8.607207
## iter  84 value 8.606859
## iter  85 value 8.605897
## iter  86 value 8.605680
## iter  87 value 8.605248
## iter  88 value 8.605128
## iter  89 value 8.605112
## iter  90 value 8.604932
## iter  91 value 8.604603
## iter  92 value 8.604494
## iter  93 value 8.604323
## iter  94 value 8.604241
## iter  95 value 8.604129
## iter  96 value 8.603898
## iter  97 value 8.603875
## iter  98 value 8.603865
## iter  99 value 8.603859
## iter 100 value 8.603859
## final  value 8.603859
## stopped after 100 iterations
## initial  value 9.237326
## iter   2 value 8.886865
## iter   3 value 8.867781
## iter   4 value 8.789210
## iter   5 value 8.689185
## iter   6 value 8.662695
## iter   7 value 8.640257
## iter   8 value 8.638800
## iter   9 value 8.638612
## iter  10 value 8.638482
## iter  11 value 8.638417
## iter  12 value 8.638105
## iter  13 value 8.637812
## iter  14 value 8.637425
```

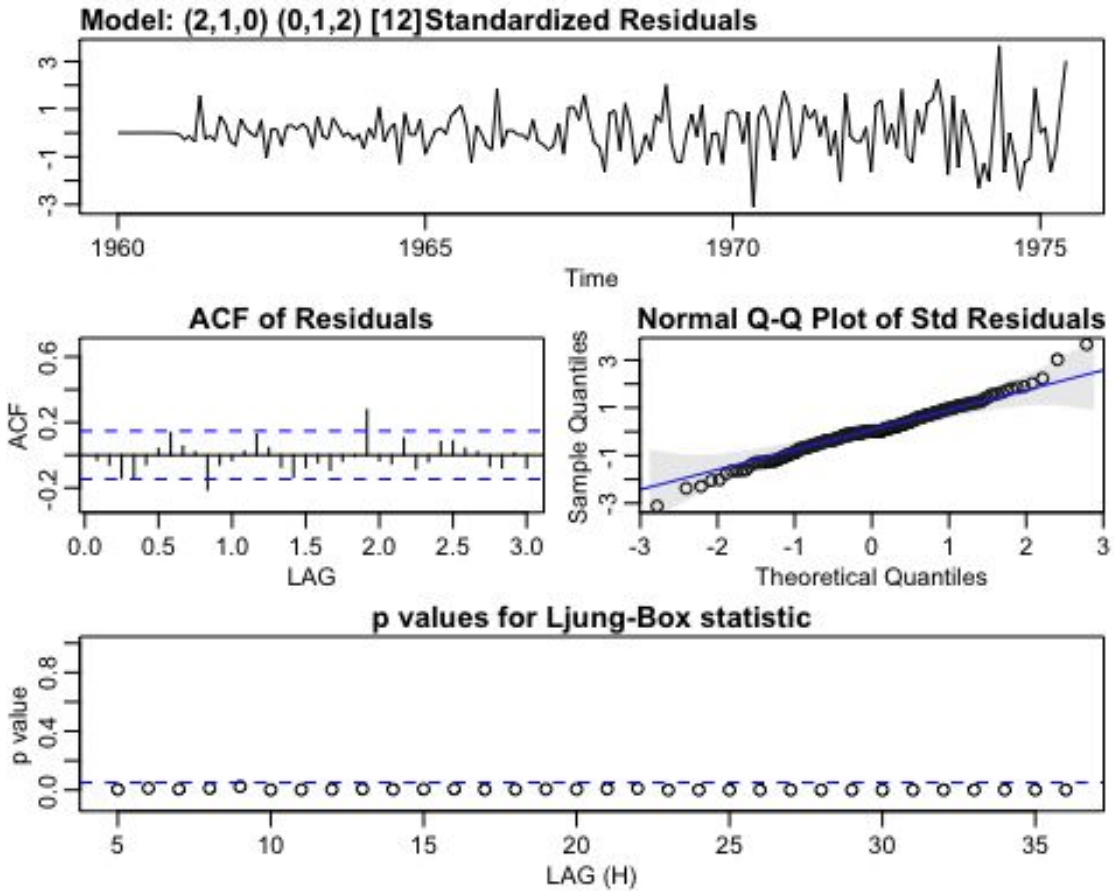
```
## iter  15 value 8.637204
## iter  16 value 8.637073
## iter  17 value 8.637061
## iter  18 value 8.637035
## iter  19 value 8.636990
## iter  20 value 8.636946
## iter  21 value 8.636921
## iter  22 value 8.636906
## iter  23 value 8.636903
## iter  24 value 8.636890
## iter  25 value 8.636881
## iter  26 value 8.636842
## iter  27 value 8.636793
## iter  28 value 8.636741
## iter  29 value 8.636709
## iter  30 value 8.636697
## iter  31 value 8.636690
## iter  32 value 8.636664
## iter  33 value 8.636609
## iter  34 value 8.636486
## iter  35 value 8.636390
## iter  36 value 8.636166
## iter  37 value 8.635719
## iter  38 value 8.635634
## iter  39 value 8.635493
## iter  40 value 8.635030
## iter  41 value 8.634643
## iter  42 value 8.634604
## iter  43 value 8.634521
## iter  44 value 8.634331
## iter  45 value 8.634058
## iter  46 value 8.633900
## iter  47 value 8.633841
## iter  48 value 8.633796
## iter  49 value 8.633727
## iter  50 value 8.633513
## iter  51 value 8.633195
## iter  52 value 8.632628
## iter  53 value 8.631078
## iter  54 value 8.629469
## iter  55 value 8.627570
## iter  56 value 8.625977
## iter  57 value 8.622632
## iter  58 value 8.618778
## iter  59 value 8.618201
## iter  60 value 8.616130
## iter  61 value 8.614755
## iter  62 value 8.612308
```

```
## iter  63 value 8.609663
## iter  64 value 8.608128
## iter  65 value 8.607504
## iter  66 value 8.606856
## iter  67 value 8.606332
## iter  68 value 8.605921
## iter  69 value 8.605899
## iter  70 value 8.605798
## iter  71 value 8.605752
## iter  72 value 8.605627
## iter  73 value 8.605338
## iter  74 value 8.604424
## iter  75 value 8.604292
## iter  76 value 8.603854
## iter  77 value 8.603820
## iter  78 value 8.603816
## iter  79 value 8.603812
## iter  80 value 8.603791
## iter  81 value 8.603785
## iter  82 value 8.603773
## iter  83 value 8.603770
## iter  84 value 8.603768
## iter  85 value 8.603768
## iter  86 value 8.603765
## iter  87 value 8.603764
## iter  88 value 8.603763
## iter  88 value 8.603763
## iter  88 value 8.603763
## final  value 8.603763
## converged
```




```
m9<-sarima(sale.ts,2,1,0,0,1,2, 12)

## initial  value 9.242919
## iter   2 value 8.941023
## iter   3 value 8.813132
## iter   4 value 8.711965
## iter   5 value 8.674127
## iter   6 value 8.657383
## iter   7 value 8.649288
## iter   8 value 8.642873
## iter   9 value 8.642318
## iter  10 value 8.642280
## iter  11 value 8.642278
## iter  12 value 8.642276
## iter  13 value 8.642275
## iter  14 value 8.642275
## iter  14 value 8.642275
## iter  14 value 8.642275
## final  value 8.642275
## converged
## initial  value 8.657354
## iter   2 value 8.657347
```

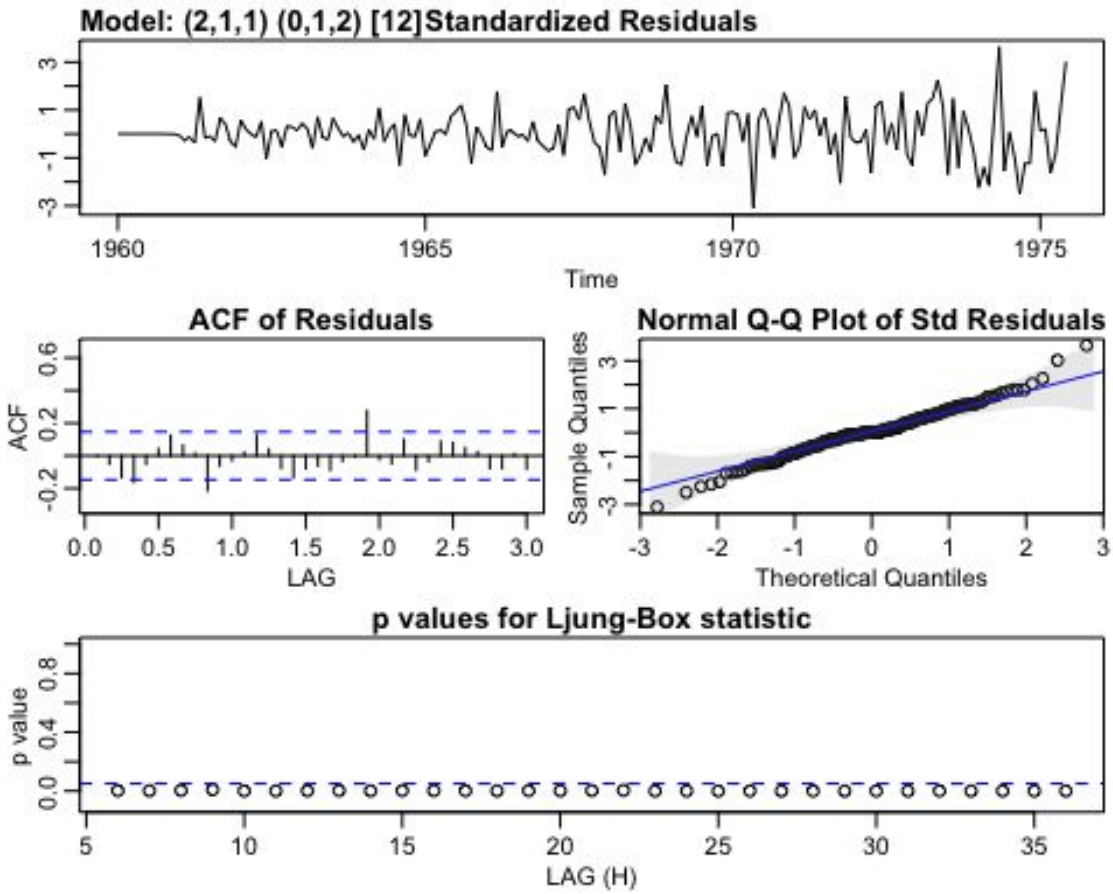
```
## iter   3 value 8.656625
## iter   4 value 8.656601
## iter   5 value 8.656455
## iter   6 value 8.656454
## iter   6 value 8.656454
## iter   6 value 8.656454
## final  value 8.656454
## converged
```




```
m10<-sarima(sale.ts,2,1,1,0,1,2, 12)
```

```
## initial  value 9.242919
## iter   2 value 8.793238
## iter   3 value 8.778226
## iter   4 value 8.697286
## iter   5 value 8.696206
## iter   6 value 8.661113
## iter   7 value 8.657557
## iter   8 value 8.653493
## iter   9 value 8.650634
## iter  10 value 8.645660
## iter  11 value 8.642914
```

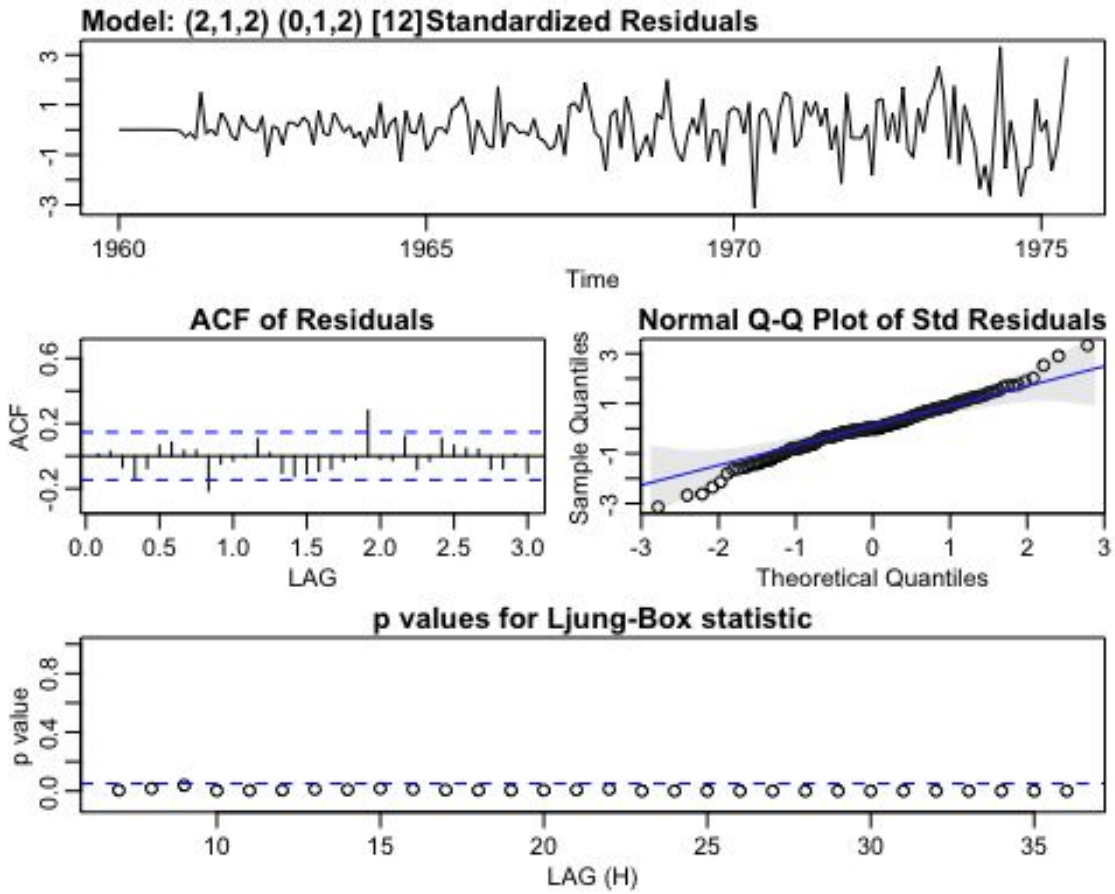
```
## iter  12 value 8.641734
## iter  13 value 8.641681
## iter  14 value 8.641643
## iter  15 value 8.641609
## iter  16 value 8.641607
## iter  17 value 8.641605
## iter  18 value 8.641603
## iter  19 value 8.641602
## iter  19 value 8.641602
## iter  19 value 8.641602
## final  value 8.641602
## converged
## initial  value 8.656472
## iter   2 value 8.655902
## iter   3 value 8.655721
## iter   4 value 8.655554
## iter   5 value 8.655523
## iter   6 value 8.655521
## iter   7 value 8.655517
## iter   8 value 8.655517
## iter   8 value 8.655517
## iter   8 value 8.655517
## final  value 8.655517
## converged
```




```
m11<-sarima(sale.ts,2,1,2,0,1,2, 12)
```

```
## initial  value 9.242919
## iter   2 value 8.791874
## iter   3 value 8.783166
## iter   4 value 8.724908
## iter   5 value 8.716702
## iter   6 value 8.680192
## iter   7 value 8.673097
## iter   8 value 8.667988
## iter   9 value 8.661105
## iter  10 value 8.654673
## iter  11 value 8.643817
## iter  12 value 8.638963
## iter  13 value 8.636535
## iter  14 value 8.635985
## iter  15 value 8.635647
## iter  16 value 8.635636
## iter  17 value 8.635609
## iter  18 value 8.635605
## iter  19 value 8.635603
## iter  20 value 8.635602
```

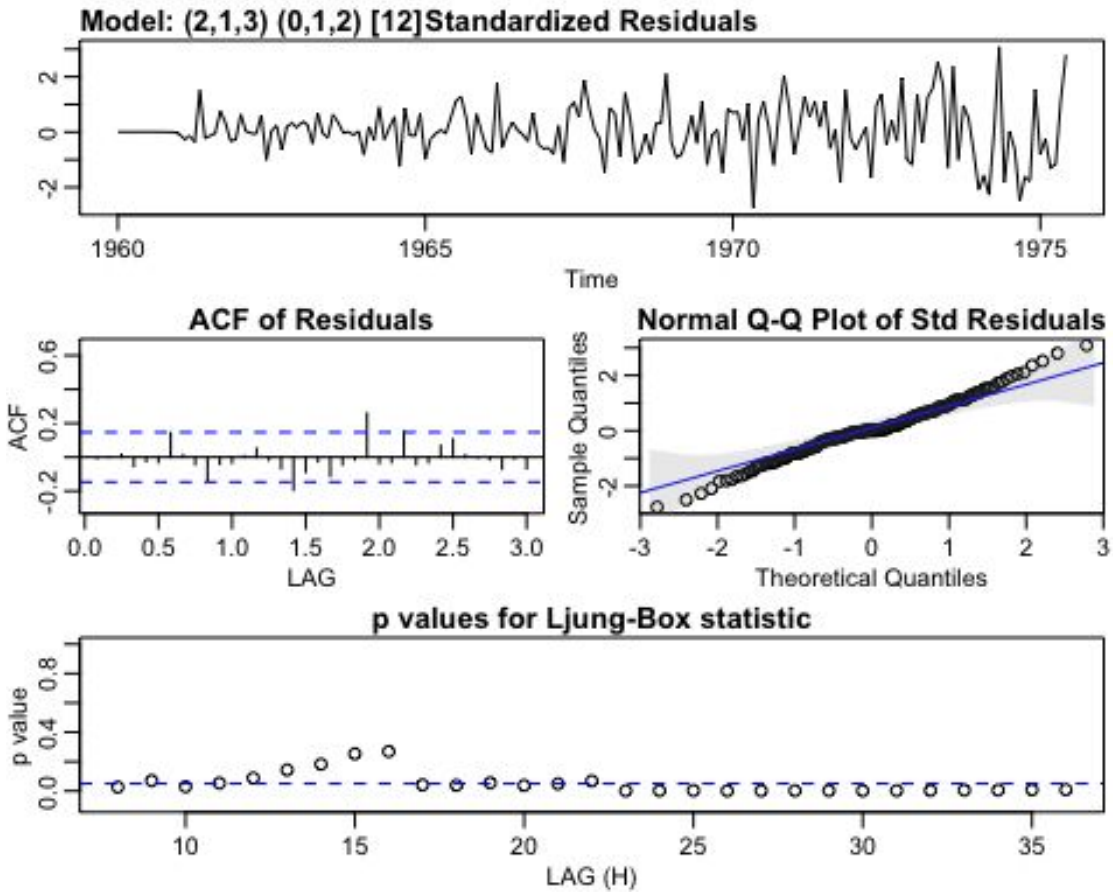
```
## iter  21 value 8.635602
## iter  21 value 8.635602
## iter  21 value 8.635602
## final  value 8.635602
## converged
## initial  value 8.649627
## iter   2 value 8.649621
## iter   3 value 8.649062
## iter   4 value 8.649030
## iter   5 value 8.648915
## iter   6 value 8.648913
## iter   7 value 8.648911
## iter   7 value 8.648911
## iter   7 value 8.648911
## final  value 8.648911
## converged
```




```
m12<-sarima(sale.ts,2,1,3,0,1,2, 12)
```

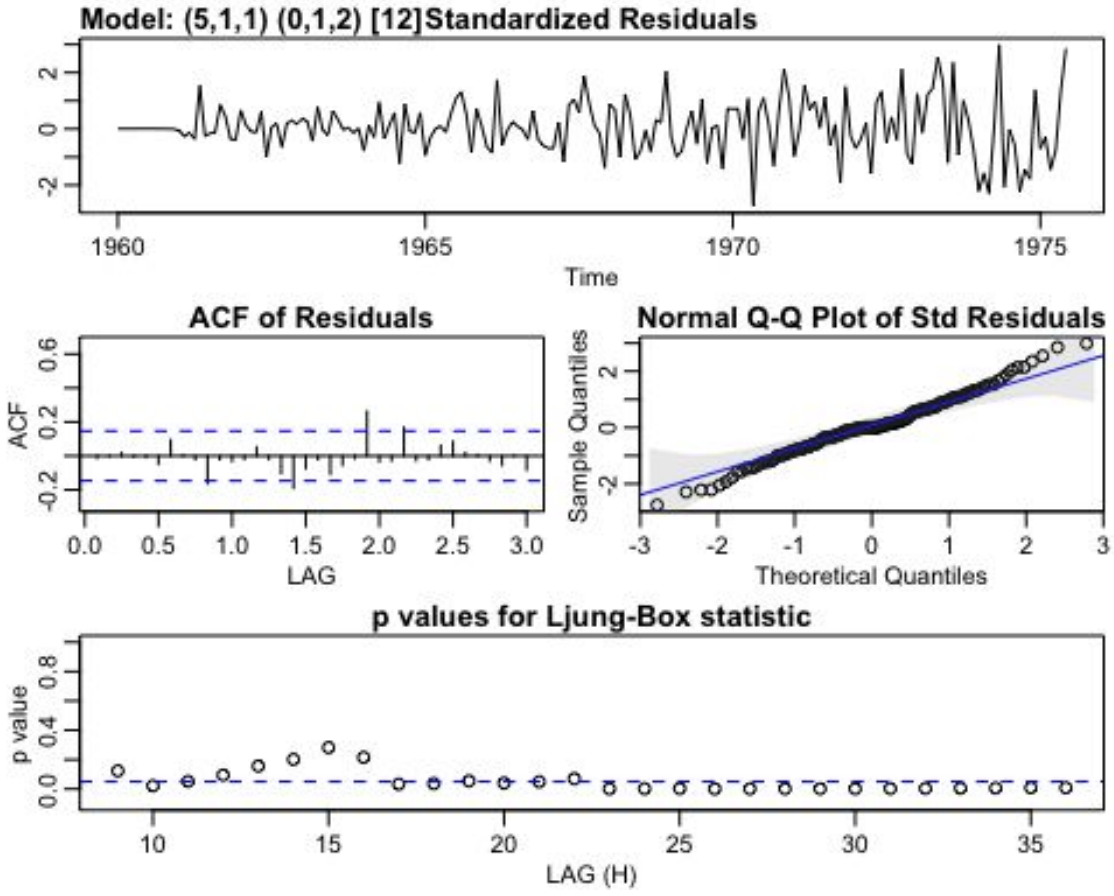
```
## initial  value 9.242919
## iter   2 value 8.910447
## iter   3 value 8.789172
```

```
## iter   4 value 8.719053
## iter   5 value 8.705509
## iter   6 value 8.692728
## iter   7 value 8.690025
## iter   8 value 8.684352
## iter   9 value 8.677711
## iter  10 value 8.657460
## iter  11 value 8.650038
## iter  12 value 8.636293
## iter  13 value 8.627297
## iter  14 value 8.624210
## iter  15 value 8.623642
## iter  16 value 8.620591
## iter  17 value 8.620289
## iter  18 value 8.619838
## iter  19 value 8.619584
## iter  20 value 8.619427
## iter  21 value 8.619406
## iter  22 value 8.619396
## iter  23 value 8.619395
## iter  23 value 8.619395
## iter  23 value 8.619395
## final  value 8.619395
## converged
## initial  value 8.632100
## iter   2 value 8.631786
## iter   3 value 8.631636
## iter   4 value 8.631588
## iter   5 value 8.631583
## iter   6 value 8.631570
## iter   7 value 8.631565
## iter   8 value 8.631562
## iter   9 value 8.631560
## iter  10 value 8.631560
## iter  10 value 8.631560
## iter  10 value 8.631560
## final  value 8.631560
## converged
```



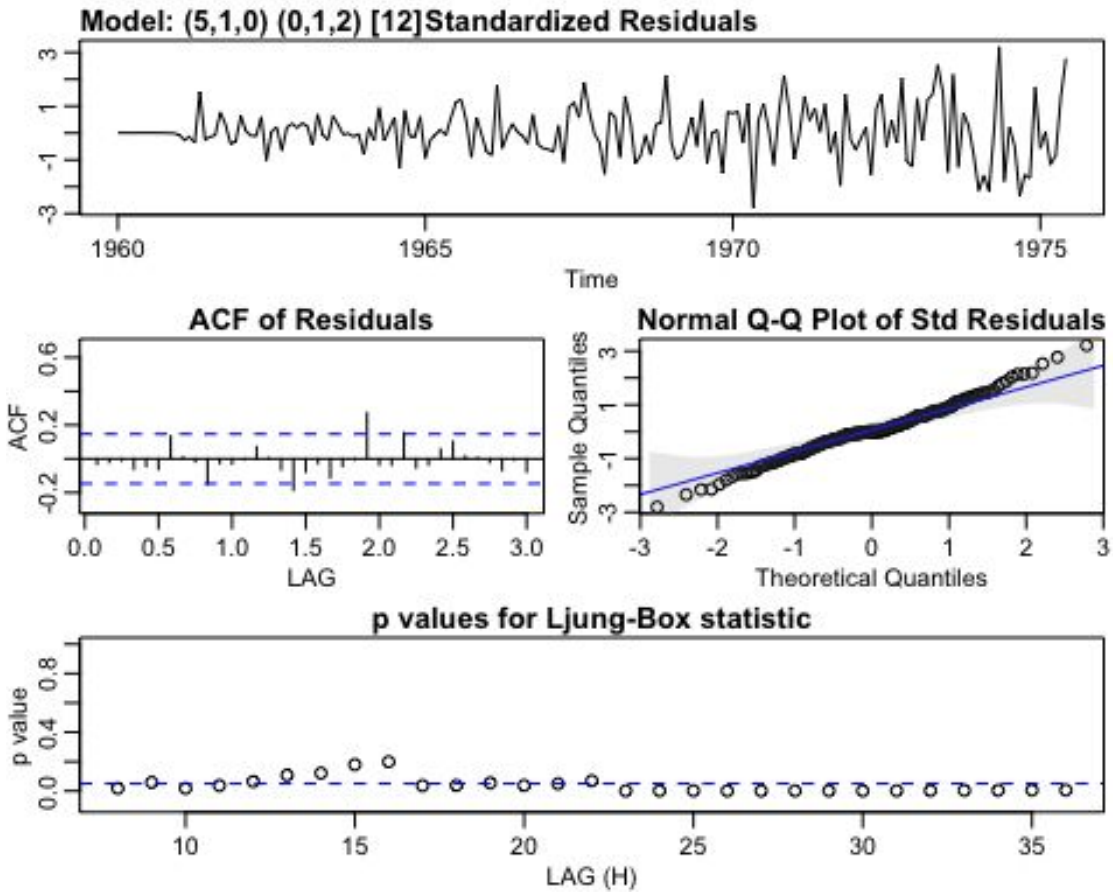

```
m13<-sarima(sale.ts,5,1,1,0,1,2, 12)

## initial  value 9.245108
## iter   2 value 8.865605
## iter   3 value 8.765047
## iter   4 value 8.680520
## iter   5 value 8.629406
## iter   6 value 8.614985
## iter   7 value 8.613879
## iter   8 value 8.613422
## iter   9 value 8.612828
## iter  10 value 8.611826
## iter  11 value 8.609399
## iter  12 value 8.608488
## iter  13 value 8.608332
## iter  14 value 8.608275
## iter  15 value 8.608269
## iter  16 value 8.608267
## iter  17 value 8.608266
## iter  18 value 8.608266
## iter  18 value 8.608266
## final  value 8.608266
```

```
## converged
## initial  value 8.633826
## iter   2 value 8.632874
## iter   3 value 8.630830
## iter   4 value 8.629372
## iter   5 value 8.628569
## iter   6 value 8.628534
## iter   7 value 8.628515
## iter   8 value 8.628500
## iter   9 value 8.628479
## iter  10 value 8.628463
## iter  11 value 8.628448
## iter  12 value 8.628445
## iter  13 value 8.628445
## iter  13 value 8.628445
## iter  13 value 8.628445
## final  value 8.628445
## converged
```




```
m14<-sarima(sale.ts,5,1,0,0,1,2, 12)
```

```
## initial  value 9.245108
## iter   2 value 9.046902
## iter   3 value 8.730011
## iter   4 value 8.690927
## iter   5 value 8.668589
## iter   6 value 8.656127
## iter   7 value 8.632885
## iter   8 value 8.621472
## iter   9 value 8.616685
## iter  10 value 8.614705
## iter  11 value 8.613921
## iter  12 value 8.613797
## iter  13 value 8.613721
## iter  14 value 8.613716
## iter  14 value 8.613716
## iter  14 value 8.613716
## final  value 8.613716
## converged
## initial  value 8.642923
## iter   2 value 8.639712
## iter   3 value 8.638346
## iter   4 value 8.636820
## iter   5 value 8.635978
## iter   6 value 8.635919
## iter   7 value 8.635907
## iter   8 value 8.635904
## iter   8 value 8.635903
## iter   8 value 8.635903
## final  value 8.635903
## converged
```



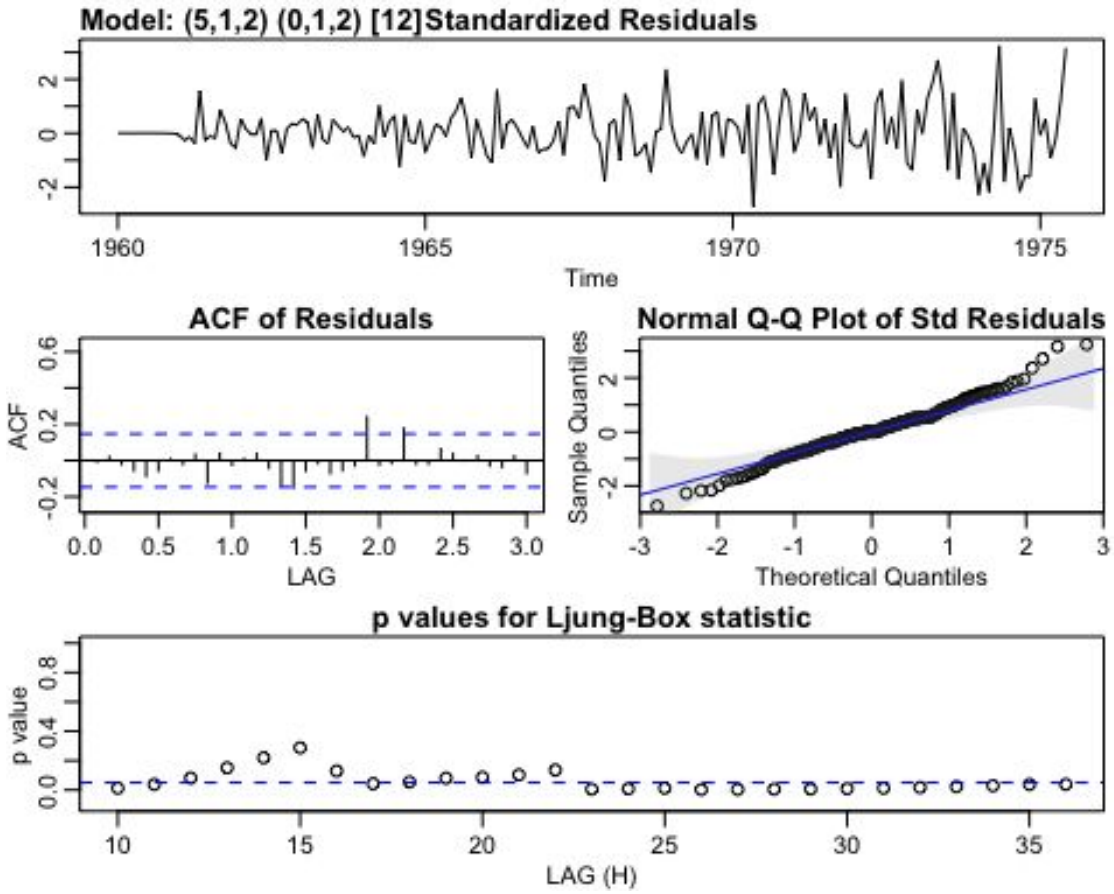

```
m15<-sarima(sale.ts,5,1,2,0,1,2, 12)
```

```
## initial  value 9.245108
## iter   2 value 8.862723
## iter   3 value 8.794420
## iter   4 value 8.678076
## iter   5 value 8.640211
## iter   6 value 8.632959
## iter   7 value 8.619429
## iter   8 value 8.613971
## iter   9 value 8.610195
## iter  10 value 8.609735
## iter  11 value 8.609009
## iter  12 value 8.608540
## iter  13 value 8.606435
## iter  14 value 8.605007
## iter  15 value 8.604508
## iter  16 value 8.604282
## iter  17 value 8.604243
## iter  18 value 8.604216
## iter  19 value 8.604201
## iter  20 value 8.604186
```

```
## iter  21 value 8.604183
## iter  22 value 8.604172
## iter  23 value 8.604158
## iter  24 value 8.604142
## iter  25 value 8.604100
## iter  26 value 8.604040
## iter  27 value 8.603842
## iter  28 value 8.603772
## iter  29 value 8.603755
## iter  30 value 8.603737
## iter  31 value 8.603717
## iter  32 value 8.603666
## iter  33 value 8.603596
## iter  34 value 8.603458
## iter  35 value 8.603316
## iter  36 value 8.603164
## iter  37 value 8.603008
## iter  38 value 8.602884
## iter  39 value 8.602799
## iter  40 value 8.602740
## iter  41 value 8.602608
## iter  42 value 8.602592
## iter  43 value 8.602591
## iter  44 value 8.602591
## iter  44 value 8.602591
## iter  44 value 8.602591
## final  value 8.602591
## converged
## initial  value 8.653232
## iter   2 value 8.651873
## iter   3 value 8.651517
## iter   4 value 8.636227
## iter   5 value 8.635676
## iter   6 value 8.634596
## iter   7 value 8.634416
## iter   8 value 8.634369
## iter   9 value 8.634278
## iter  10 value 8.633981

## Warning in log(s2): NaNs produced

## iter  11 value 8.633476
## iter  12 value 8.633275
## iter  13 value 8.632853
## iter  14 value 8.632688
## iter  15 value 8.632292
## iter  16 value 8.631177
## iter  17 value 8.631094
```

```
## iter  18 value 8.629242
## iter  19 value 8.628854
## iter  20 value 8.628698
## iter  21 value 8.628629
## iter  22 value 8.628295
## iter  23 value 8.627968
## iter  24 value 8.627296
## iter  25 value 8.626366
## iter  26 value 8.625366
## iter  27 value 8.623853
## iter  28 value 8.623062
## iter  29 value 8.621798
## iter  30 value 8.620791
## iter  31 value 8.619571
## iter  32 value 8.619253
## iter  33 value 8.618795
## iter  34 value 8.618423
## iter  35 value 8.617792
## iter  36 value 8.617149
## iter  37 value 8.616683
## iter  38 value 8.616151
## iter  39 value 8.615712
## iter  40 value 8.615569
## iter  41 value 8.615314
## iter  42 value 8.615157
## iter  43 value 8.615096
## iter  44 value 8.614994
## iter  45 value 8.614841
## iter  46 value 8.614577
## iter  47 value 8.614387
## iter  48 value 8.612858

## Warning in log(s2): NaNs produced

## iter  49 value 8.612211
## iter  50 value 8.611433
## iter  51 value 8.610851
## iter  52 value 8.610107
## iter  53 value 8.609808
## iter  54 value 8.609494
## iter  55 value 8.609424
## iter  56 value 8.609413
## iter  57 value 8.609405
## iter  58 value 8.609389
## iter  59 value 8.609384
## iter  60 value 8.609381
## iter  61 value 8.609381
## iter  61 value 8.609381
```

```
## iter  61 value 8.609381
## final  value 8.609381
## converged
```

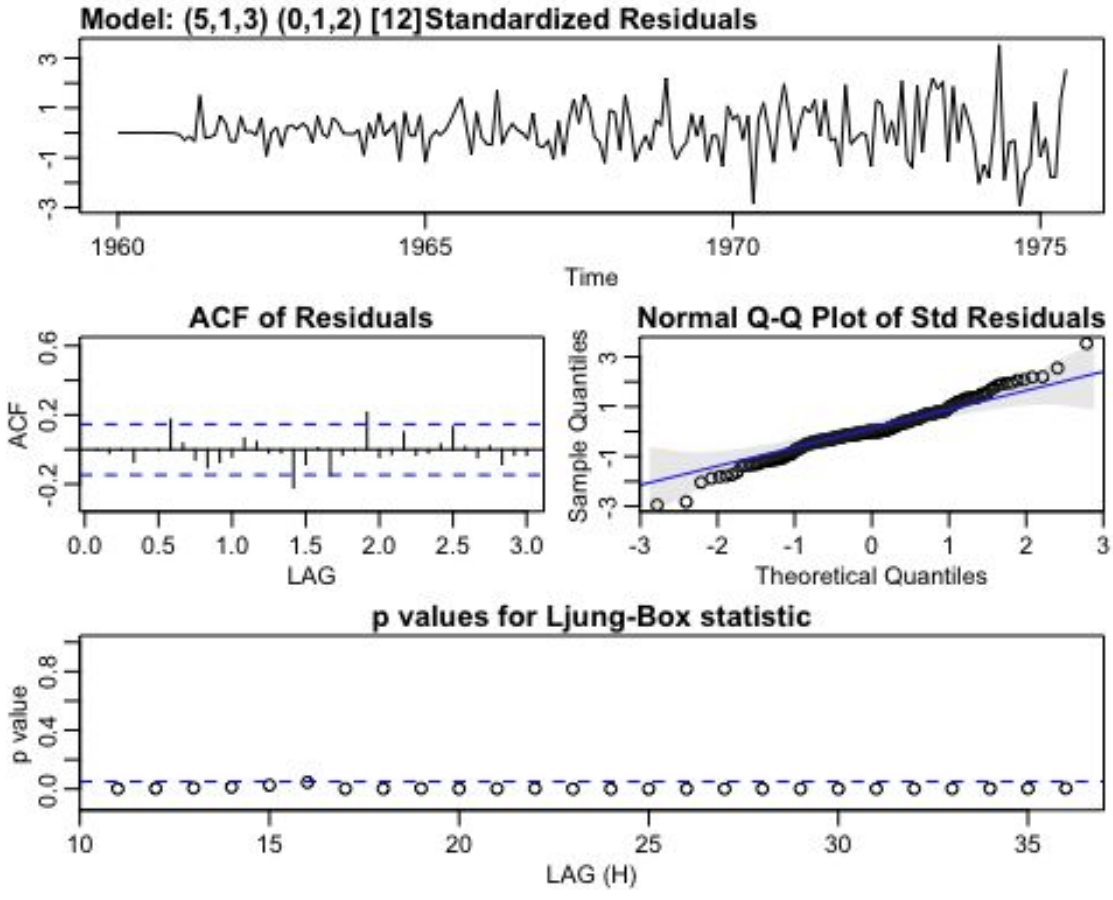



```
m16<-sarima(sale.ts,5,1,3,0,1,2, 12)

## initial  value 9.245108
## iter   2 value 8.981566
## iter   3 value 8.854635
## iter   4 value 8.752699
## iter   5 value 8.658767
## iter   6 value 8.626537
## iter   7 value 8.617043
## iter   8 value 8.614332
## iter   9 value 8.613122
## iter  10 value 8.612521
## iter  11 value 8.611463
## iter  12 value 8.607228
## iter  13 value 8.605898
## iter  14 value 8.605047
## iter  15 value 8.604464
## iter  16 value 8.603594
```

```
## iter  17 value 8.602906
## iter  18 value 8.601984
## iter  19 value 8.601638
## iter  20 value 8.601201
## iter  21 value 8.600854
## iter  22 value 8.600410
## iter  23 value 8.600331
## iter  24 value 8.600197
## iter  25 value 8.600008
## iter  26 value 8.599851
## iter  27 value 8.599721
## iter  28 value 8.599624
## iter  29 value 8.599560
## iter  30 value 8.599496
## iter  31 value 8.599349
## iter  32 value 8.599125
## iter  33 value 8.598893
## iter  34 value 8.598771
## iter  35 value 8.598729
## iter  36 value 8.598718
## iter  37 value 8.598713
## iter  38 value 8.598713
## iter  38 value 8.598713
## iter  38 value 8.598713
## final  value 8.598713
## converged
## initial  value 8.654722
## iter   2 value 8.652998
## iter   3 value 8.650926
## iter   4 value 8.641787
## iter   5 value 8.640716
## iter   6 value 8.637448
## iter   7 value 8.634745
## iter   8 value 8.633592
## iter   9 value 8.633552
## iter  10 value 8.633333
## iter  11 value 8.633176
## iter  12 value 8.632676
## iter  13 value 8.632246
## iter  14 value 8.631546
## iter  15 value 8.631141
## iter  16 value 8.631077
## iter  17 value 8.631049
## iter  18 value 8.631035
## iter  19 value 8.630998
## iter  20 value 8.630951
## iter  21 value 8.630812
## iter  22 value 8.630611
```

```
## iter  23 value 8.630044
## iter  24 value 8.629326
## iter  25 value 8.628501
## iter  26 value 8.627645
## iter  27 value 8.626405
## iter  28 value 8.625547
## iter  29 value 8.624871
## iter  30 value 8.624470
## iter  31 value 8.621798
## iter  32 value 8.619806
## iter  33 value 8.613525
## iter  34 value 8.613113
## iter  35 value 8.612366
## iter  36 value 8.610847
## iter  37 value 8.609290
## iter  38 value 8.607672
## iter  39 value 8.605033
## iter  40 value 8.602569
## iter  41 value 8.599442
## iter  42 value 8.599094
## iter  43 value 8.598285
## iter  44 value 8.598199
## iter  45 value 8.597767
## iter  46 value 8.597597
## iter  47 value 8.597580
## iter  48 value 8.597552
## iter  49 value 8.597529
## iter  50 value 8.597520
## iter  51 value 8.597504
## iter  52 value 8.597476
## iter  53 value 8.597413
## iter  54 value 8.597372
## iter  55 value 8.597358
## iter  56 value 8.597346
## iter  57 value 8.597316
## iter  58 value 8.597295
## iter  59 value 8.597164
## iter  60 value 8.597118
## iter  61 value 8.596994
## iter  62 value 8.596821
## iter  63 value 8.596702
## iter  64 value 8.596448
## iter  65 value 8.596354
## iter  66 value 8.596275
## iter  67 value 8.596241
## iter  68 value 8.596226
## iter  69 value 8.596204
## iter  70 value 8.596189
```

```
## iter  71 value 8.596139
## iter  72 value 8.596126
## iter  73 value 8.596123
## iter  74 value 8.596120
## iter  75 value 8.596117
## iter  76 value 8.596115
## iter  77 value 8.596114
## iter  78 value 8.596112
## iter  79 value 8.596108
## iter  80 value 8.596108
## iter  81 value 8.596107
## iter  81 value 8.596107
## iter  82 value 8.596107
## iter  82 value 8.596107
## iter  82 value 8.596107
## final  value 8.596107
## converged
```




```
m1$AIC
```

```
## [1] 19.00323
```

```
m2$AIC
## [1] 19.00961
m3$AIC
## [1] 19.00937
m4$AIC
## [1] 18.98406
m5$AIC
## [1] 18.99541
m6$AIC
## [1] 18.99867
m7$AIC
## [1] 18.98334
m8$AIC
## [1] 18.95573
m9$AIC
## [1] 19.00047
m10$AIC
## [1] 19.00957
m11$AIC
## [1] 19.00802
m12$AIC
## [1] 18.98626
m13$AIC
## [1] 18.99128
m14$AIC
## [1] 18.99443
m15$AIC
```

```
## [1] 18.9663
```

```
m16$AIC
```

```
## [1] 18.95221
```

The lowest AIC scores were 18.95 for model 16, 18.97 for model 15, 18.896 for model 12, 18.956 for model 8, 18.983 for model 7 and 18.984 for model 4. By the law of parsimony, we want to choose the model with the fewest parameters, so we decide on model 4 with 6 parameters.

```
fit<-ar(y12diff1, method=c("yule-walker"))
fit
```

```
##
## Call:
## ar(x = y12diff1, method = c("yule-walker"))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -1.0188  -0.7546  -0.2594  -0.2537  -0.2838  -0.1902  -0.0484   0.0679
##       9       10       11       12       13       14       15       16
##  0.0269  -0.1622  -0.2175  -0.4782  -0.4267  -0.2803  -0.0555  -0.1315
##      17       18       19       20       21       22
## -0.2695  -0.2860  -0.2177  -0.1484  -0.2094  -0.2992
##
## Order selected 22  sigma^2 estimated as  0.001189
```

```
fit2<-ar(sale.ts, method=c("yule-walker"))
fit2
```

```
##
## Call:
## ar(x = sale.ts, method = c("yule-walker"))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.6845   0.0558   0.2039  -0.2306   0.2127  -0.1671   0.0950  -0.0096
##       9       10       11       12       13       14       15
##  0.0735   0.0511   0.0542   0.3525  -0.2504   0.0431  -0.2080
##
## Order selected 15  sigma^2 estimated as  171790949
```

```
var(y12)
```

```
## [1] 0.001672437
```

```
library(qpcR)
```

```
## Loading required package: minpack.lm
```

```
## Loading required package: rgl

## Warning: package 'rgl' was built under R version 3.5.2

## Warning in rgl.init(initValue, onlyNULL): RGL: unable to open X11 display

## Warning: 'rgl_init' failed, running with rgl.useNULL = TRUE

## Loading required package: robustbase

## Warning: package 'robustbase' was built under R version 3.5.2

## Loading required package: Matrix
```

```
aiccs<-matrix(NA, nr = 6, nc = 6)
dimnames(aiccs) = list(p = 0:5, q = 0:5)
for(p in 0:5)
{
  for(q in 0:5)
  {
    aiccs[p+1, q+1] = AICc(arima(y1diff12,
order=c(p,1,q),seasonal=list(order=c(0,1,1),period=12), method = "ML"))
  }
}
```

```
## 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 arima(y1diff12, order = c(p, 1, q), seasonal = list(order =
## c(0, : possible convergence problem: optim gave code = 1

## Warning in arima(y1diff12, order = c(p, 1, q), seasonal = list(order =
## c(0, : possible convergence problem: optim gave code = 1
```

```
aiccs
```

```
##    q
## p           0         1         2         3         4         5
##   0 -212.2309 -385.4586 -492.1460 -518.2044 -517.3989 -530.3066
##   1 -303.1831 -451.1364 -513.1029 -516.1826 -522.3501 -529.3542
##   2 -443.8764 -533.3171 -532.2694 -532.8583 -534.1040 -531.9329
##   3 -473.5126 -531.9361 -537.2547 -535.1498 -532.9829 -531.8539
```

```
##   4 -481.2003 -530.8261 -535.1511 -532.9071 -531.2271 -529.9422
##   5 -489.4757 -532.8947 -533.5236 -531.7675 -545.6945 -543.2074
```

```
model4<- arima(sale.ts, order = c(2, 1, 3), seasonal = list(order = c(0, 1,
1), period = 12))
model4
```

```
##
## Call:
## arima(x = sale.ts, order = c(2, 1, 3), seasonal = list(order = c(0, 1, 1),
period = 12))
##
## Coefficients:
##           ar1      ar2      ma1     ma2      ma3     sma1
##       -1.0298  -0.8367  -0.0118  0.0217  -0.3765  -0.6736
## s.e.   0.0761   0.0864   0.1290  0.0979   0.1427   0.0797
##
## sigma^2 estimated as 30151984:  log likelihood = -1739.53,  aic = 3493.07
```

```
Box.test(model4$residuals, lag = 14, type = c("Box-Pierce"), fitdf = 6)
```

```
##
##  Box-Pierce test
##
## data:  model4$residuals
## X-squared = 10.051, df = 8, p-value = 0.2614
```

```
Box.test(model4$residuals, lag =14, type = c("Ljung-Box"), fitdf = 6)
```

```
##
##  Box-Ljung test
##
## data:  model4$residuals
## X-squared = 10.639, df = 8, p-value = 0.223
```

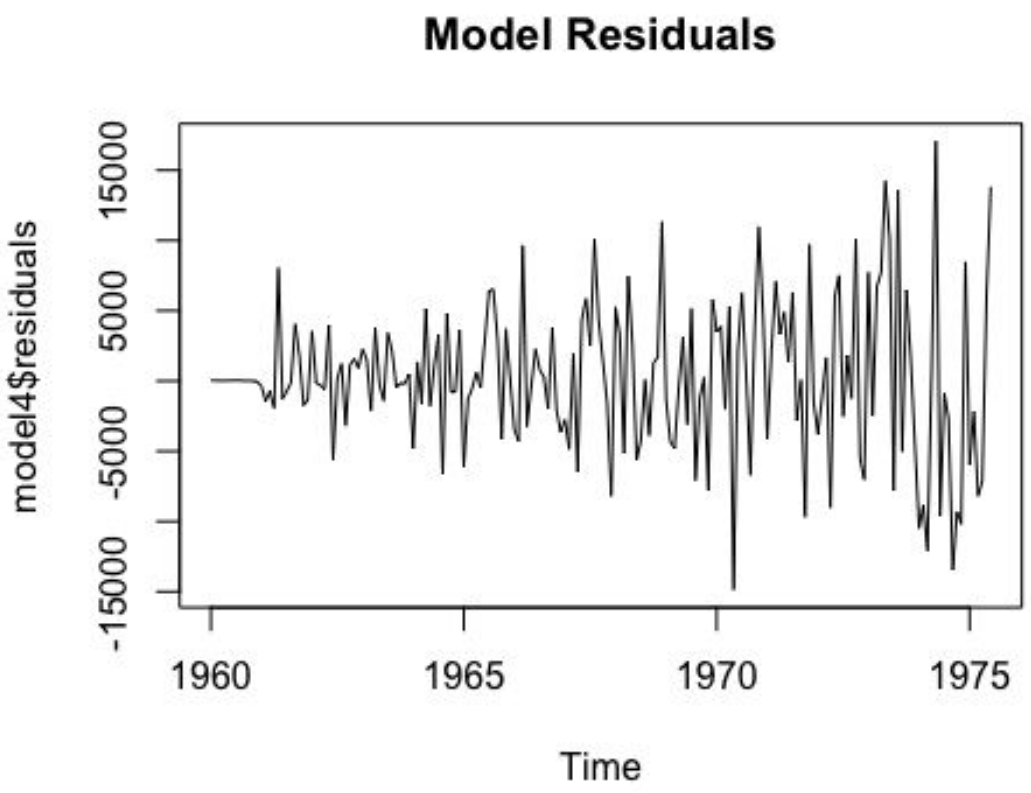
```
Box.test((model4$residuals)^2, lag =14, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  (model4$residuals)^2
## X-squared = 142.16, df = 14, p-value < 2.2e-16
```

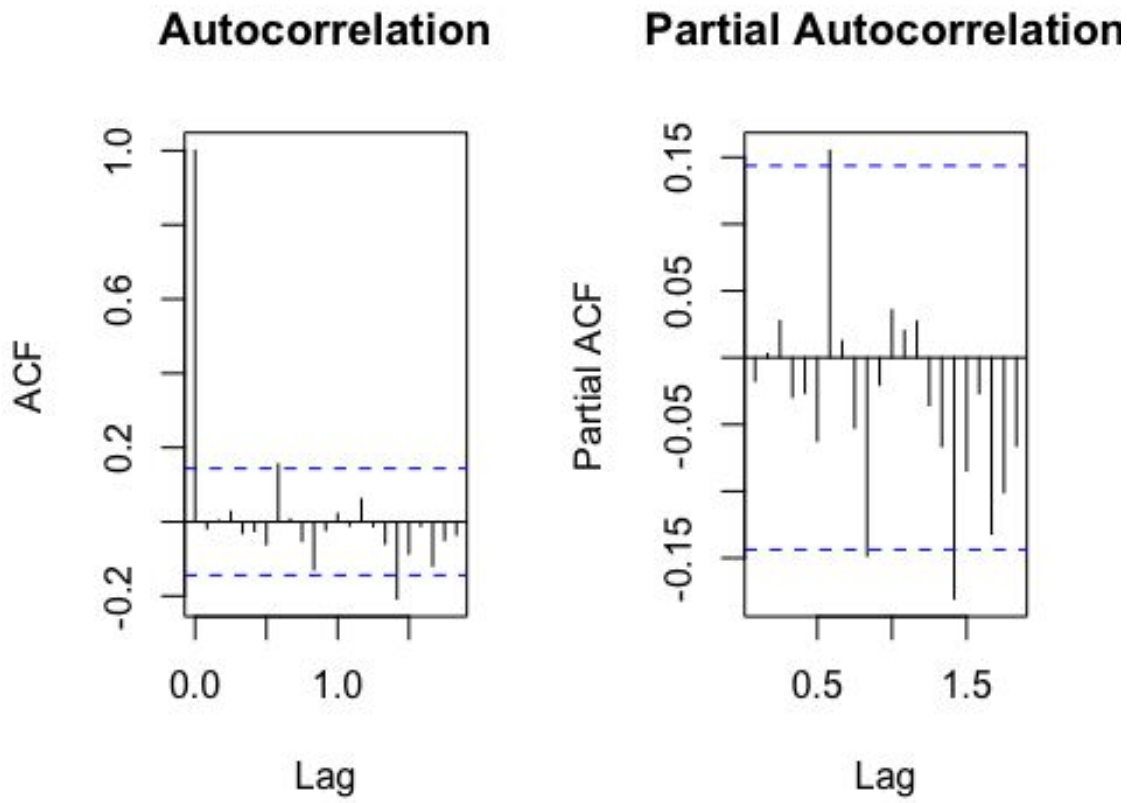
```
shapiro.test(model4$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4$residuals
## W = 0.9889, p-value = 0.1559
```

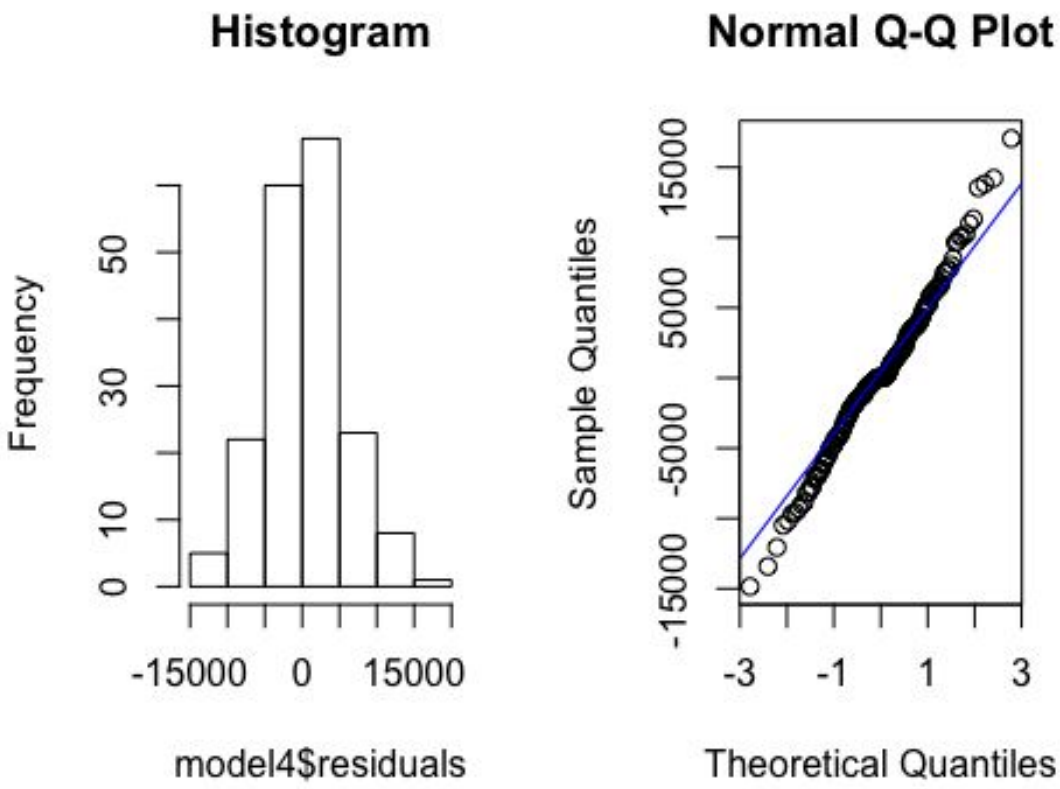
```r
ts.plot(model4$residuals, main="Model Residuals")
```




```r
op = par(mfrow = c(1,2))
acf(model4$residuals,main = "Autocorrelation")
pacf(model4$residuals,main = "Partial Autocorrelation")
```




```r
hist(model4$residuals,main = "Histogram")
qqnorm(model4$residuals)
qqline(model4$residuals, col = "blue")
```




```
model16<- arima(sale.ts, order = c(5, 1, 3), seasonal = list(order = c(0, 1,
2), period = 12))
model16
```

```
##
## Call:
## arima(x = sale.ts, order = c(5, 1, 3), seasonal = list(order = c(0, 1, 2),
period = 12))
##
## Coefficients:
##           ar1      ar2     ar3     ar4     ar5     ma1      ma2      ma3
##       -1.2543  -0.9349  0.4016  0.5186  0.2936  0.2722  -0.0395  -0.8923
## s.e.   0.2473   0.5164  0.6865  0.4541  0.1866  0.2375   0.2614   0.1720
##          sma1     sma2
##       -0.6252  -0.0525
## s.e.   0.1011   0.0906
##
## sigma^2 estimated as 26969235:  log likelihood = -1732.6,  aic = 3487.21
```

```
Box.test(model16$residuals, lag = 14, type = c("Box-Pierce"), fitdf = 10)
```

```
##
##  Box-Pierce test
##
## data:  model16$residuals
## X-squared = 12.013, df = 4, p-value = 0.01725
```

```
Box.test(model16$residuals, lag =14, type = c("Ljung-Box"), fitdf = 10)
```

```
##
##  Box-Ljung test
##
## data:  model16$residuals
## X-squared = 12.72, df = 4, p-value = 0.01273
```

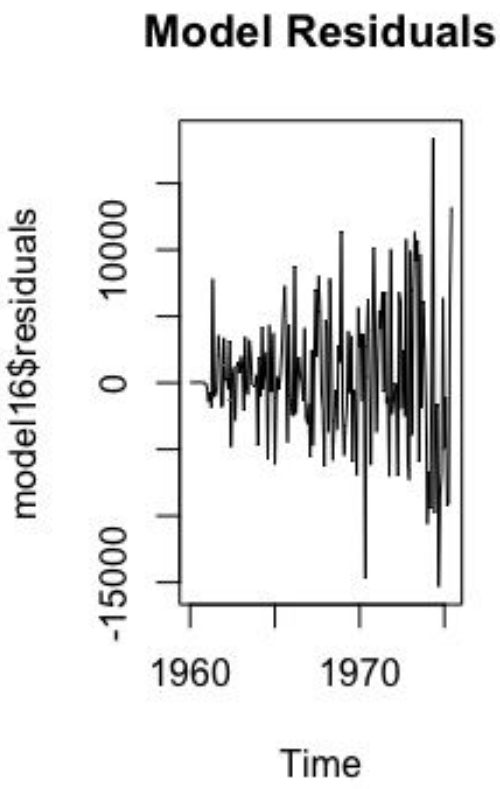
```
Box.test((model16$residuals)^2, lag =14, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  (model16$residuals)^2
## X-squared = 114.05, df = 14, p-value < 2.2e-16
```

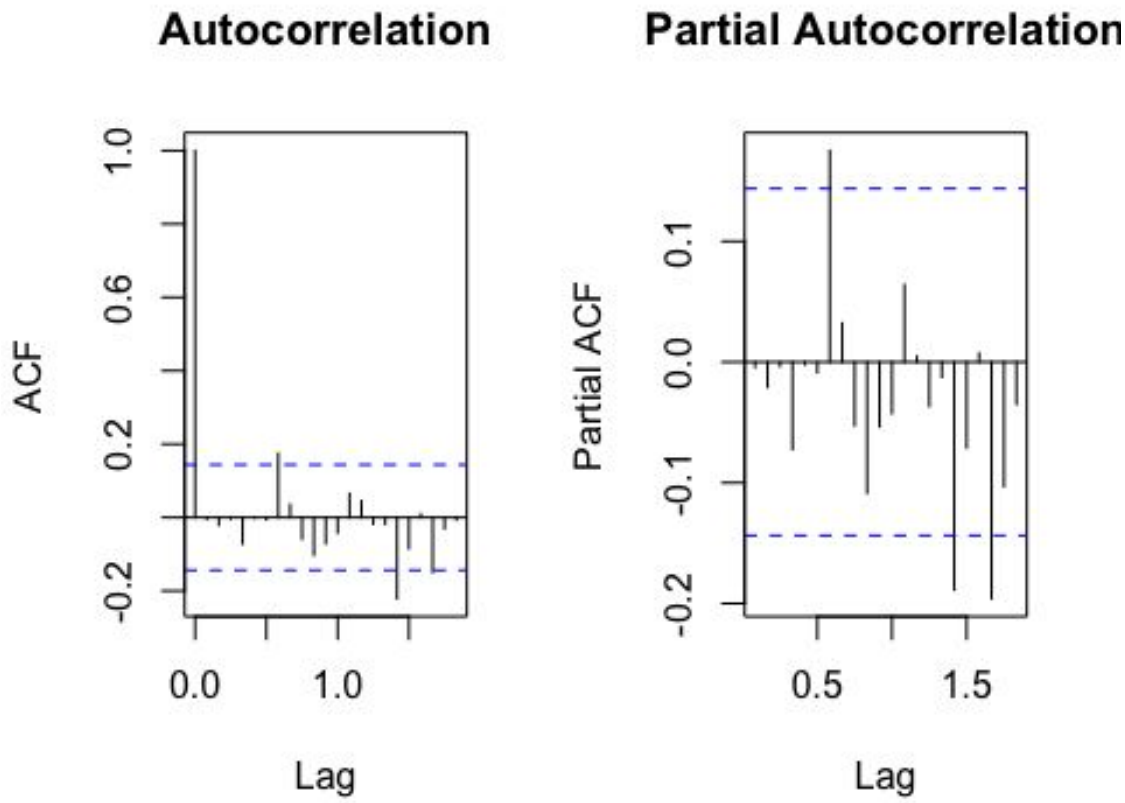
```
shapiro.test(model16$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model16$residuals
## W = 0.98061, p-value = 0.01099
```

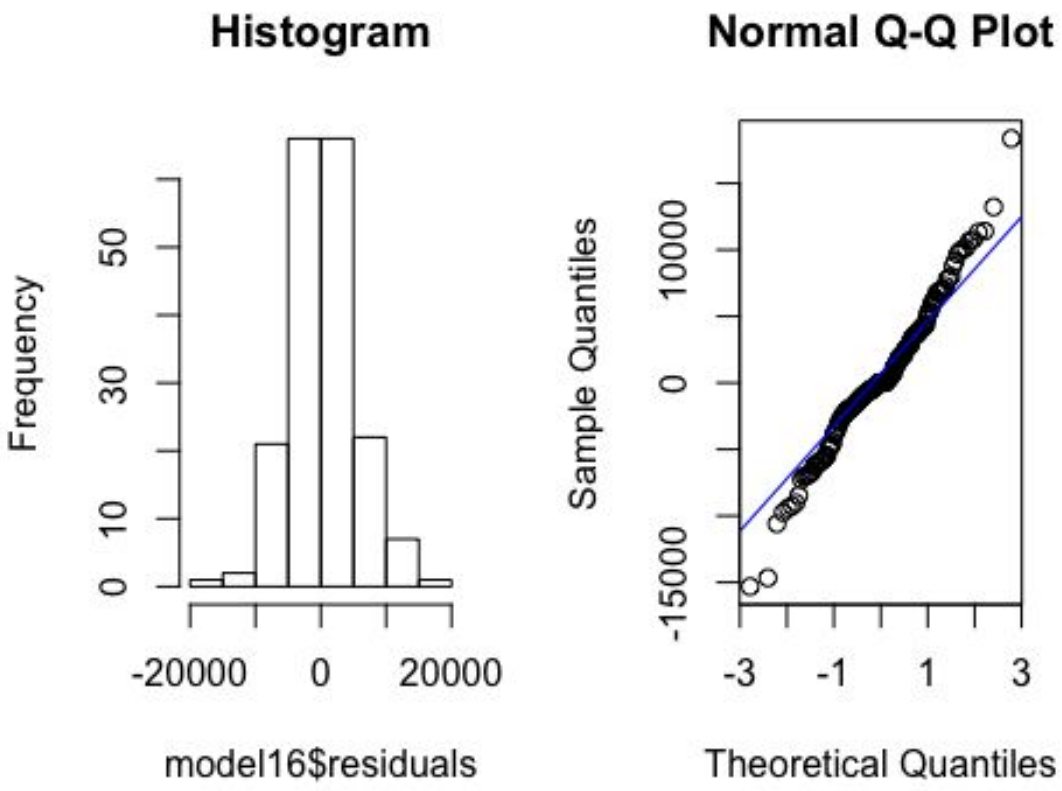
```
ts.plot(model16$residuals, main="Model Residuals")
op = par(mfrow = c(1,2))
```




```r
acf(model16$residuals,main = "Autocorrelation")
pacf(model16$residuals,main = "Partial Autocorrelation")
```

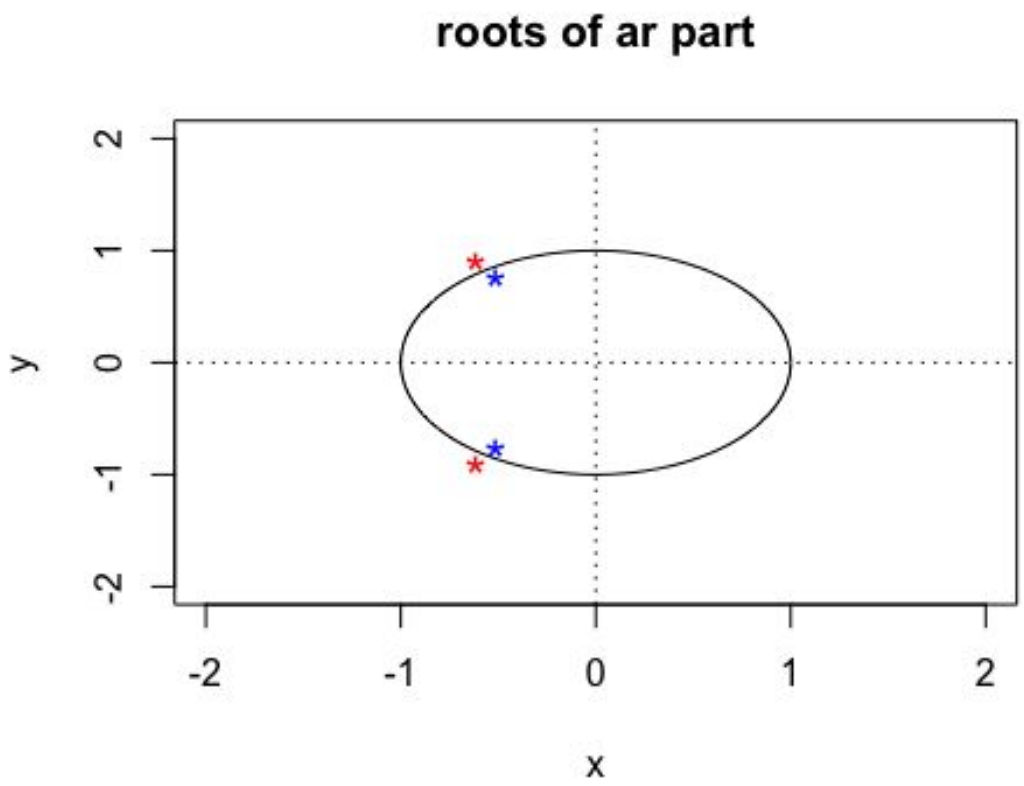



```r
hist(model16$residuals,main = "Histogram")
qqnorm(model16$residuals)
qqline(model16$residuals, col = "blue")
```




```
plot.roots <- function(ar.roots=NULL, ma.roots=NULL, size=2, angles=FALSE,
special=NULL,
sqecial=NULL,my.pch=1,first.col="blue",second.col="red",main=NULL)
{xylims <- c(-size,size)
      omegas <- seq(0,2*pi,pi/500)
      temp <- exp(complex(real=rep(0,length(omegas)),imag=omegas))

plot(Re(temp),Im(temp),typ="l",xlab="x",ylab="y",xlim=xylims,ylim=xylims,main
=main)
      abline(v=0,lty="dotted")
      abline(h=0,lty="dotted")
      if(!is.null(ar.roots))
        {
          points(Re(1/ar.roots),Im(1/ar.roots),col=first.col,pch=my.pch)
          points(Re(ar.roots),Im(ar.roots),col=second.col,pch=my.pch)
        }
      if(!is.null(ma.roots))
        {
          points(Re(1/ma.roots),Im(1/ma.roots),pch="*",cex=1.5,col=first.col)
          points(Re(ma.roots),Im(ma.roots),pch="*",cex=1.5,col=second.col)
        }
      if(angles)
```

```
        {
          if(!is.null(ar.roots))
            {
              abline(a=0,b=Im(ar.roots[1])/Re(ar.roots[1]),lty="dotted")
              abline(a=0,b=Im(ar.roots[2])/Re(ar.roots[2]),lty="dotted")
            }
          if(!is.null(ma.roots))
            {
              sapply(1:length(ma.roots), function(j)
abline(a=0,b=Im(ma.roots[j])/Re(ma.roots[j]),lty="dotted"))
            }
        }
      if(!is.null(special))
        {
          lines(Re(special),Im(special),lwd=2)
        }
      if(!is.null(sqecial))
        {
          lines(Re(sqecial),Im(sqecial),lwd=2)
        }
        }

model<- arima(sale.ts, order = c(2, 1, 3), seasonal = list(order = c(0, 1,
1), period = 12))
model

##
## Call:
## arima(x = sale.ts, order = c(2, 1, 3), seasonal = list(order = c(0, 1, 1),
period = 12))
##
## Coefficients:
##           ar1      ar2      ma1     ma2      ma3     sma1
##       -1.0298  -0.8367  -0.0118  0.0217  -0.3765  -0.6736
## s.e.   0.0761   0.0864   0.1290  0.0979   0.1427   0.0797
##
## sigma^2 estimated as 30151984:  log likelihood = -1739.53,  aic = 3493.07

plot.roots(NULL,polyroot(c(1, 1.0298, 0.8367)), main="roots of ar part")
```

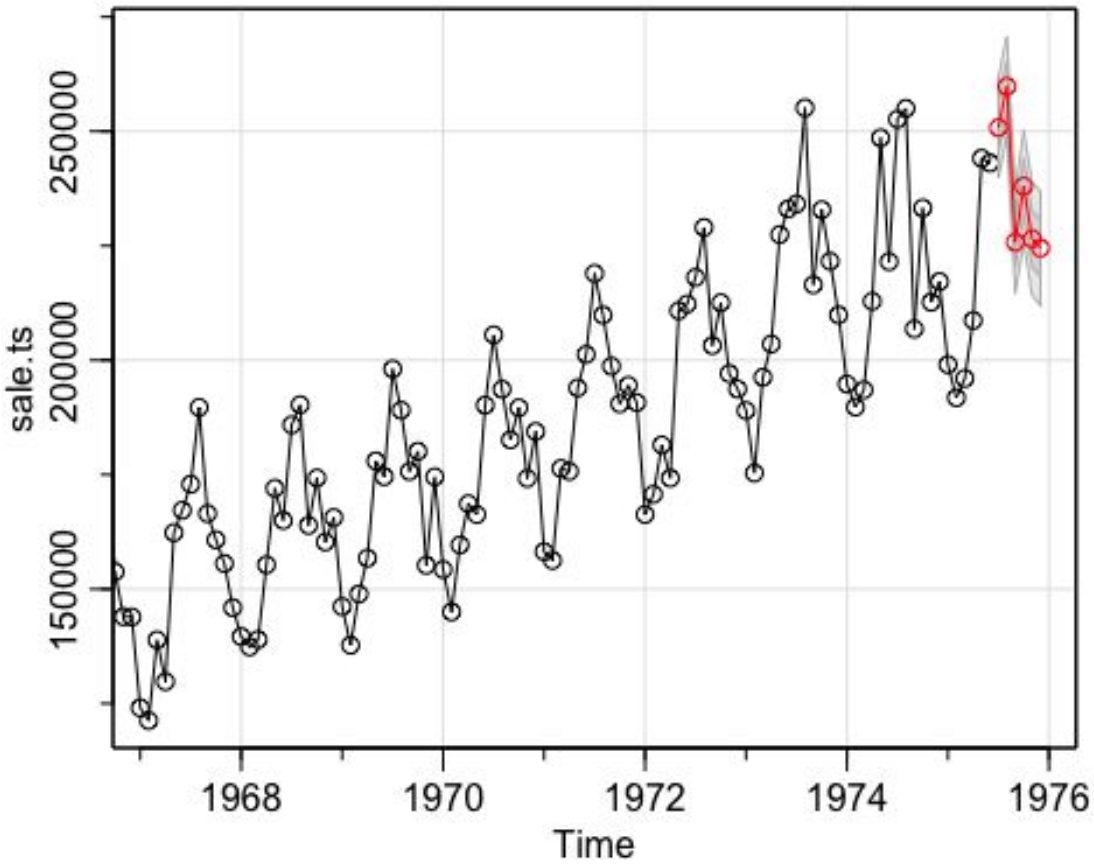



```
plot.roots(NULL,polyroot(c(1, -0.0118, 0.0217, -0.3765)), main="roots of ma
part")
```

```
plot.roots(NULL,polyroot(c(1, -0.6736)), main="roots of seasonal ma part")
```

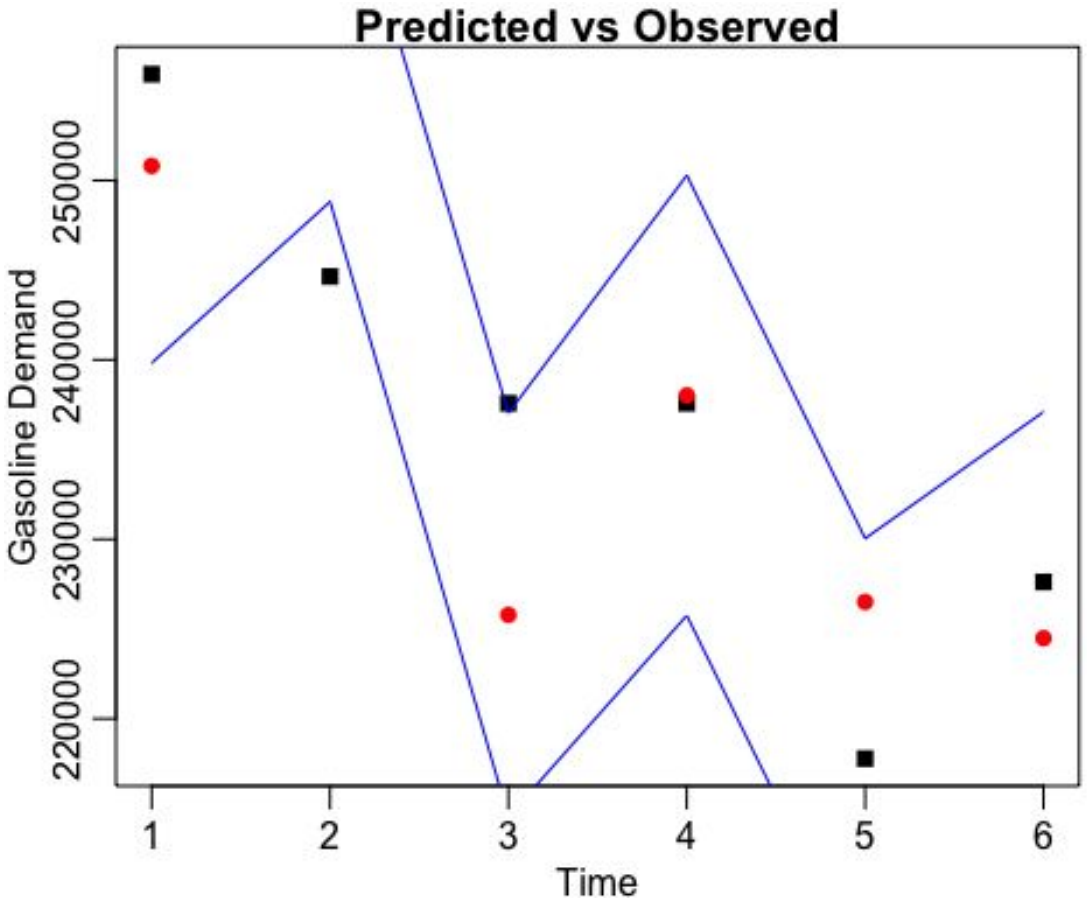
All the roots are outside the unit circle, this passes the root test, meaning the model is causal and invertible. All the coefficients are significant.

```
sarima.for(sale.ts, 6, 2, 1, 3, 0, 1, 1, 12)
```




```
## $pred
##           Jul      Aug      Sep      Oct      Nov      Dec
## 1975 250804.0 259798.2 225782.5 238013.9 226507.4 224485.9
##
## $se
##           Jul      Aug      Sep      Oct      Nov      Dec
## 1975 5491.092 5495.828 5622.363 6132.156 6132.255 6305.060
```

```
x<-c(1,2,3,4,5,6)
Observed <- c(255918, 244642, 237579, 237579, 217775, 227621)
Upper<-c(239821.816, 248806.544, 214537.774, 225749.588, 205510.49,
211875.78)
Lower<-c(261786.184, 270789.856, 237027.226, 250278.212, 230039.51,
237096.02)
Predicted<-c(250804.0, 259798.2, 225782.5, 238013.9, 226507.4, 224485.9)
plot(x,Observed, xlab = "Time", ylab="Gasoline Demand", main="Predicted vs
Observed", pch=15, col="black")
lines(x, Upper, pch=16, col="blue")
lines(x, Lower, pch=16, col="blue")
points(x, Predicted, pch=16, col="red")
```


Predicted vs Observed
Gasoline Demand
220000
230000
240000
250000
1
2
3
4
5
6
Time